Gothic Bookshop Milestone 2

Alex Hoffman, Alex Hong, Arien Leigh, Joey Ilagan, Lina Nayvelt

Changes/Updates

Since our first proposal we've brainstormed two more features: a recommendation system based on common authors/genres, and a way to associate books with classes at Duke. With that second feature, users could search for classes and find required books (rather than searching in a syllabus). We added requirements around professors' ability to require or suggest material for their classes. We specified how professors will go through the book requirement process for their classes.

A feature where students and professors can request books or classes be added to the store's web system was also added to the intended final product.

Added abilities and specifications around how students are able to review books on the site were incorporated. Students are now able to rank their peer's reviews of material along with edit their own reviews. Books are now searchable by their ISBN number along with the previously specified title and author.

We have also modified the basic feature descriptions from the Mini-Amazon project. Specifically, we reduced the stakeholders to 3 main groups: students, professors, and admins. We also changed certain bookstore specific features such as allowing the user to list which class they are purchasing a book from in order to allow for us to display relevant data to our bookstore owners and the professors.

Proposal

We plan to create a Gothic Bookstore with 3 user types- Users, Professors, and Admins. Each of these users will have distinct features/functions of the site available to them. Some features overlap between user types.

Basic	**Weight**
Users are able to login and create an account as a User, Professor, or Admin - Users account information should be stored and associated with purchases the user makes (students/professors), comments the user makes, and classes the user teaches (professor only).	8%
Search bar is functional and all book information is able to be displayed - Search results show a summary of books returned for the search. - Users can search by book title, book authors, and book class association.	24%

- After searching for a book and getting results, users (students/professors) can click on a book and be taken to a page with more details on that product. On that page users (students/professors) should be able to add the book to their cart.	
Students are able to leave reviews on website and filter - Reviews have a title, description, rating, class the reviewing student used the book for, and a recommendation of whether other students should purchase the product or not. - Reviews will show up on a books page. - Past reviews a student has given will appear on their homepage. - Reviews can be edited. - Reviews can be upvoted/downvoted by other students.	24%
Professors and Admin are able to see detailed user statistics - Admin should be able to view how different books are doing in their store. - Admin should be able to adjust prices of merchandise. Admin should be able to restock items when they get a shipment. - Admin should be able to keep track of shipments to them from suppliers. - Admin should be able to be able to view book buying trends (sorted by certain classes a book was bought for or by the book itself) - Admin should be able to view order history. - Professors should be able to view student reviews for all books. Reviews for books they require or recommend should be sorted out to some capability for professors to be able to see. - Professors should be able to view purchasing history information for books associated with classes they teach.	24%
Cart is functional and appropriately keeps track of inventory - Create a cart and save for later function. - Purchasing accurately updates inventory and user balance. - Purchasing reliably creates an order which can be viewed by the admin or the user who made the purchase at a later time.	20%
Bonus	
Recommendation System based on common authors/genre	10%
Organize books into reading lists	10%

Summary of Progress

Created a GitLab repo and granted access to all members so that we are all able to collaborate and edit our project. We have a requirements.txt file that uniformly specifies our dependencies.

We have chosen Flask as the engine to run our database and are planning to couple it with html files to create the visuals for each page.

Created an Entity-Relationship diagram with a schema outlining the layout that our Gothic-Bookstore will take on. Mapped a user flow - for our design scheme there are a total of 17 pages that we plan to use. Please see below for links to view that work.

Tasks to be Completed
Further work includes finding/creating the databases that we will use for books and their inventory as well as creating the flask templates and html pages that will contain the content.

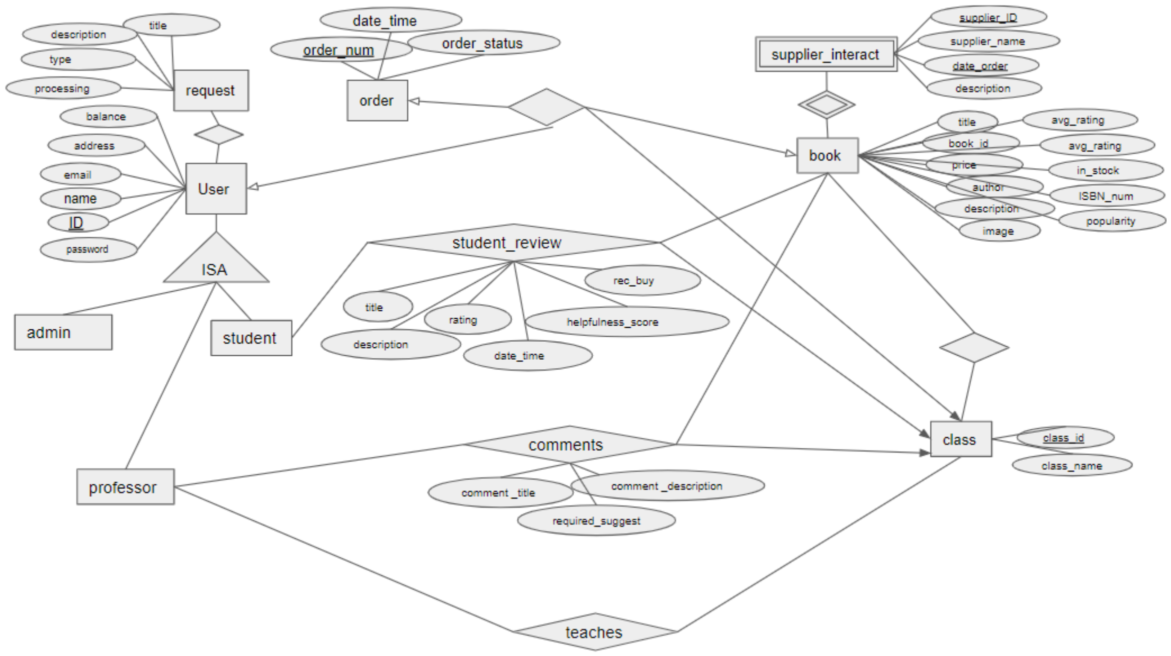
Task	Status	Notes
Finding/Creating Databases	In progress	- may need to use Python or Sheets script to automatically generate additional data
Create mock data that will allow for initial testing of code.	In progress	- Initial mock data has been created, further mock data will be made as features get added (requesting books, reviewing books, etc.)
Creating Flask and HTML pages with contents	Not started	
Writing code that searches data tables to return books in a book search.	Not started	
Writing code that creates a new student_review/professor_comment when a user requests that function	Not started	
Write code that allows admin to search orders	Not started	
Write code that allows admin to view purchasing trends overtime	Not started	
Write code that allows admin to create new classes or add new books to the system.	Not started	
Write code allowing admin to update a book price or number in	Not started	

Task	Status	Notes
stock.		
Write code that allows admin to create a teaching relationship between a teacher and a class	Not started	
Write code that depicts what should take place when a order is placed	Not started	
Write code that allows admin to update order status	Not started	
Write code that allows order information to be displayed to the student/professor users	Not started	
Write code that allows for the generation of a request from a student/professor to a admin official. Write code that allows for an admin official to review that request.	Not started	
Write code that allows admin to add information about their interactions with suppliers.	Not started	
Write code that allows students/professors to edit/delete their reviews.	Not started	
Write code for a functioning chart that processes order such that inventory and user balance is accurately updated.	Not started	
Write code to create a user homepage that allows user access to their user type specific functions.	Not started	
Write code to display search results. Write code to allow searching for books/orders to take a user to that book/order specific	Not started	

Task	Status	Notes
page to continue with their search activity.		
Write code to display the cart as a user updates it.	Not started	
Write code to allow for users to update their personal information.	Not started	

ERD & Database Design

Link to diagrams: ER Diagram



student(<u>ID</u>, password, name, email, address, balance)
Prof(<u>ID</u>, password, name, email, address, balance)
Admin(<u>ID</u>, password, name, email, address, balance)

order(<u>order_num</u>, date_time, order_status)
supplier_interact(<u>book_id</u>, supplier, date_supplier_order, description)
book(<u>book_ID</u>, title, author, description, image, price, popularity_score, ISBN_number, avg_rating, in_stock)

class(<u>class_id</u>, class_name)

student_review(<u>student_ID</u>, class_ID, <u>book_ID</u>, rating, title, description, recommend_binary, date_time, helpfulness_score)
prof_book(<u>prof_ID</u>, class_ID, <u>book_ID</u>, req_suggest, comment_title, comment_description)

Assumptions
The attributes associated with user, orders, classes and books were directly interpreted from what was required to be displayed or searched by in the features documents.

The attributes clearly associated with qualities that represent a relationship between two entities were associated with those relationships. A comment is meant to be associated with a teacher, the comments they make should be accessible by them through their homepage. Comments by professors should also be associated with a specific book. Thus commenting is a relationship between entities and the attributes of the comment (title, description, required or suggested) were associated with the relationship. Since a single professor should be able to make many comments and a single book can have multiple comments it wouldn't have made sense to associate the comment title and description with a single book. Comments are also meant to refer to a specific class or no class at all, thus at most one arrow is drawn from the comment to a class. The same logic applied for student reviews and their associated attributes.

A "teaches" relationship exists between teacher and class so that teachers with the privilege of requiring or suggesting a book can be kept track of.

Books are in a relationship with classes, as books need to be searchable by the classes they are recommended for.

Orders have users who bought books associated, a book bought in the order, and a maximum of one class the book was bought for (we intend to keep track of which classes result in the most purchase for admin and professors to be able to view). Therefore a relationship between orders, users, classes, and books was created to allow for order searching and information display as described in our complete features document.

UI/UX/User Flow
Link to UI mockups: Gothic Bookshop Page Design