CST-391 Milestone Music Store

Kaya Nelson

College of Science, Engineering, and Technology, Grand Canyon

University Course Number: CST-91

Professor Estey

3/5/2025

GitHub Link:

Kdeshun/CST-391

Introduction

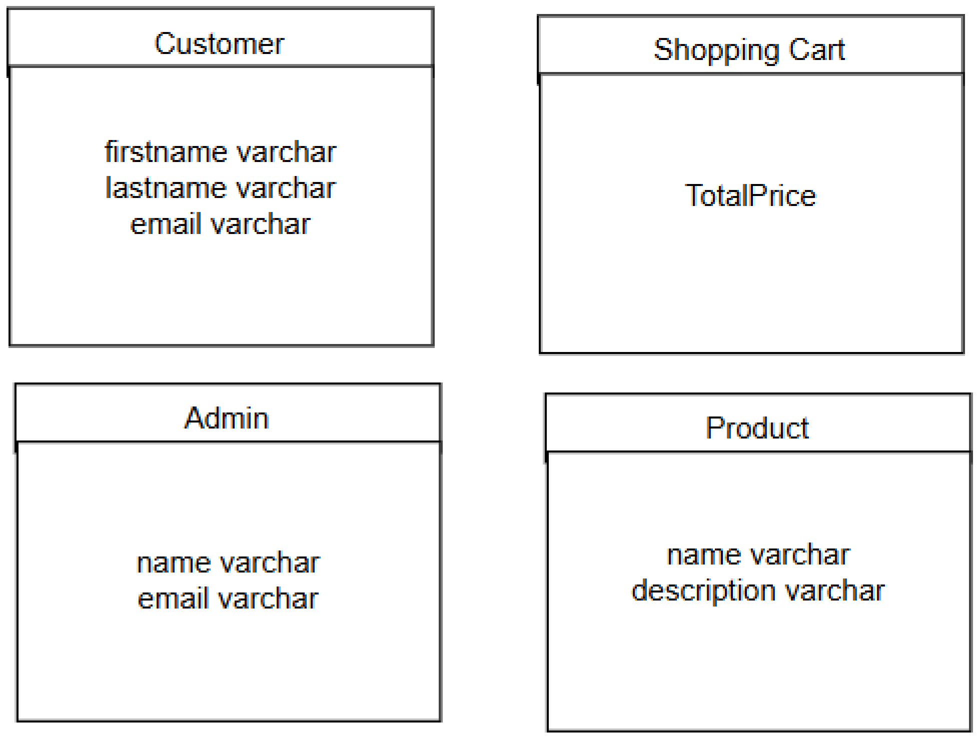
- For this milestone, I am envisioning the development of a comprehensive music store that offers an extensive range of products. This store will feature an array of musical instruments, including popular options like guitars and pianos, catering to musicians of all skill levels. In addition, it will provide essential accessories such as guitar picks, strings, and other gear that enhance the playing experience. Furthermore, the store will stock a diverse selection of music media, including CDs and DVDs, ensuring that customers can enjoy both classic and contemporary music collections. This project aims to create a one-stop-shop for music enthusiasts, combining quality products with an engaging shopping experience.
- The application will enable users to effortlessly browse and manage a wide range of products through an intuitive back-end and front-end system. It will integrate functionalities for listing, creating, reading, updating, and deleting products, ensuring a comprehensive management experience. This milestone will serve as a demonstration of the effective application of both back-end and front-end technologies. It will involve the use of databases for data storage, REST APIs for seamless communication between the client and server, and user interfaces designed for a smooth and engaging user experience. Overall, this project will showcase the integration of various technologies to create a fully functional music store application.

Functionality Requirements

1. As a customer, I want the ability to browse through a comprehensive list of available products so that I can explore the inventory of the music store. This feature will allow me to easily view and discover various musical instruments, accessories, and media, enhancing my shopping experience and helping me find items that meet my needs.
2. As a customer, I want to access detailed information about a specific product so that I can learn more about its features, specifications, and pricing before making a purchase decision. This will help me ensure that the product meets my needs and expectations, allowing for a more informed buying experience.

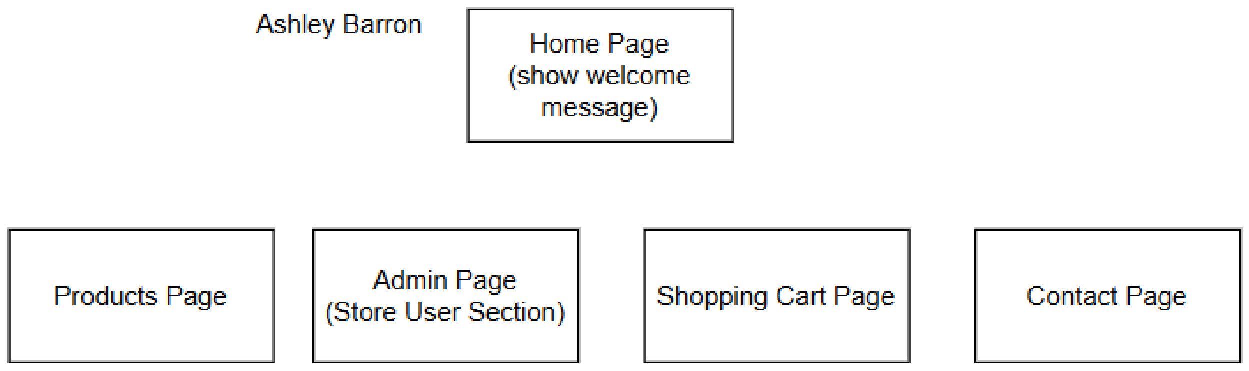
3. As a customer, I want the capability to search for specific items using keywords or categories. This feature will enable me to quickly locate particular products within the music store's inventory, streamlining my shopping experience and making it easier to find exactly what I'm looking for without having to browse through all available options.
4. As a store user, I want the ability to delete products from the inventory that are no longer available. This functionality will help maintain an accurate and up-to-date catalog, ensuring that customers only see products that are currently in stock.
5. As a store user, I want to have the capability to add new products to the store's inventory. This will allow me to expand the range of offerings and keep the product selection fresh and appealing to customers.
6. As a store user, I want to update the details of existing products in the inventory. This feature will enable me to ensure that all product information, such as descriptions, prices, and specifications, remains accurate and reflects any changes, providing customers with the most reliable information.

ER Diagram



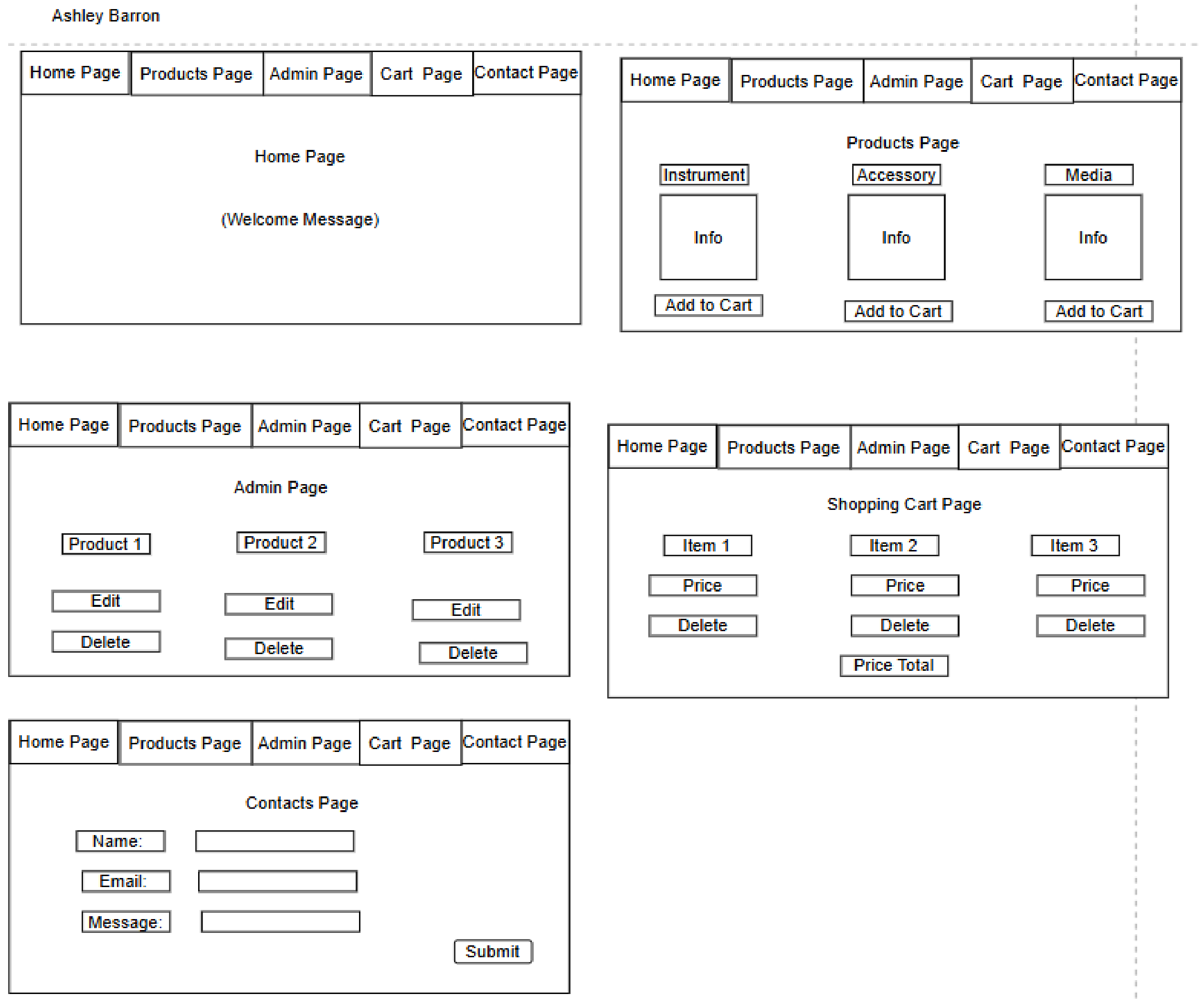
UI Sitemap Diagram

Ashley Barron


Home Page (show welcome message)
Products Page
Admin Page (Store User Section)
Shopping Cart Page
Contact Page

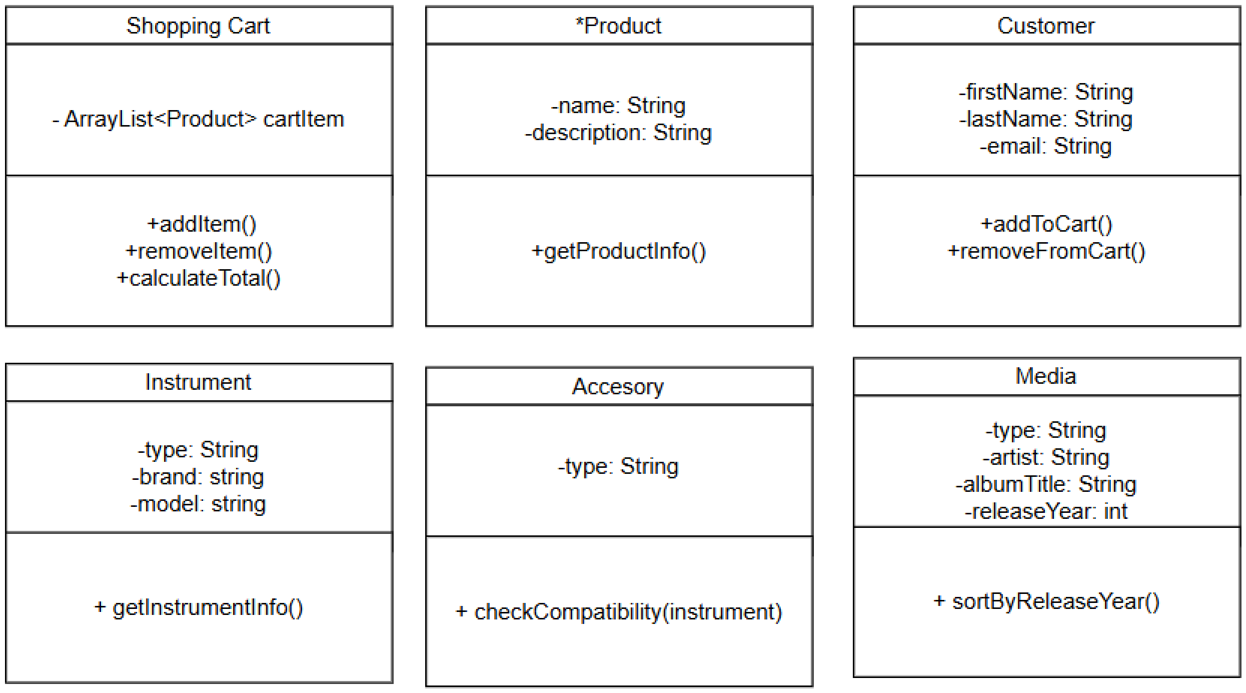
UI Wireframe

Ashley Barron



UML Classes

Ashley Barron



Risks

1. **Database Design Issues**:
   - **Challenge**: As the project progresses, the need for modifications to the database schema may arise. This could be due to evolving business requirements, such as adding new product categories, incorporating user reviews, or adjusting pricing structures. If not designed with adaptability in mind, these changes could lead to significant restructuring, data migration issues, or loss of data integrity.
   - **Mitigation Strategy**:
     - **Modular Design**: Adopt a modular approach in database design by using techniques like normalization and creating separate tables for categories, products, and user data. This structure will facilitate easier modifications and scalability.
     - **Version Control**: Implement a version control system for database schemas. This will allow for tracking changes and rolling back in case of issues. Additionally, document all changes thoroughly to provide a clear history of the database evolution.
2. **Time Management**:

- **Challenge**: Juggling multiple classes, projects, and milestones can lead to overwhelming workloads, especially when deadlines coincide or unforeseen circumstances arise. This can result in subpar quality of work or missed deadlines, affecting the overall project timeline and educational commitments.
- **Mitigation Strategy**:
  - **Prioritized Task Lists**: Create a detailed task list categorized by urgency and importance. Use tools like the Eisenhower Matrix to distinguish between what is urgent and important, enabling better focus on critical tasks.
  - **Time Blocking**: Implement time blocking techniques to dedicate specific hours to project work, ensuring uninterrupted focus. Additionally, set aside buffer time for unexpected delays or challenges.
  - **Regular Check-ins**: Schedule weekly check-ins to assess progress and adjust timelines as needed. This can help identify potential bottlenecks early and allow for timely interventions.

3. **Technical Debt**:
   - **Challenge**: The pressure to meet deadlines may lead developers to take shortcuts, resulting in technical debt characterized by poorly structured code, lack of documentation, or insufficient testing. This can create long-term maintenance challenges and reduce the overall quality of the application.
   - **Mitigation Strategy**:
     - **Incremental Development**: Embrace an agile development approach, focusing on delivering incremental features with regular feedback loops. This allows for continuous improvement and helps maintain code quality.
     - **Code Reviews and Pair Programming**: Establish a culture of code reviews and pair programming. This encourages collaboration, knowledge sharing, and adherence to coding standards, reducing the likelihood of messy code.

    - **Technical Debt Tracking**: Create a dedicated backlog for technical debt items and prioritize addressing them in future sprints. This ensures that technical debt is not ignored and is systematically managed over time.

Conclusion

By proactively addressing these challenges with tailored strategies, the project can be guided toward a successful outcome while maintaining high standards of quality, efficiency, and adaptability. This approach not only enhances the immediate project experience but also fosters long-term skills and practices beneficial for future endeavors.