SLEEPOVERS FOR LIFE

Design Document

University of British Columbia Okanagan
COSC 499: Capstone - Summer 2019
Developed by: Everton Smith, Joshua Henderson, Trevor Richard

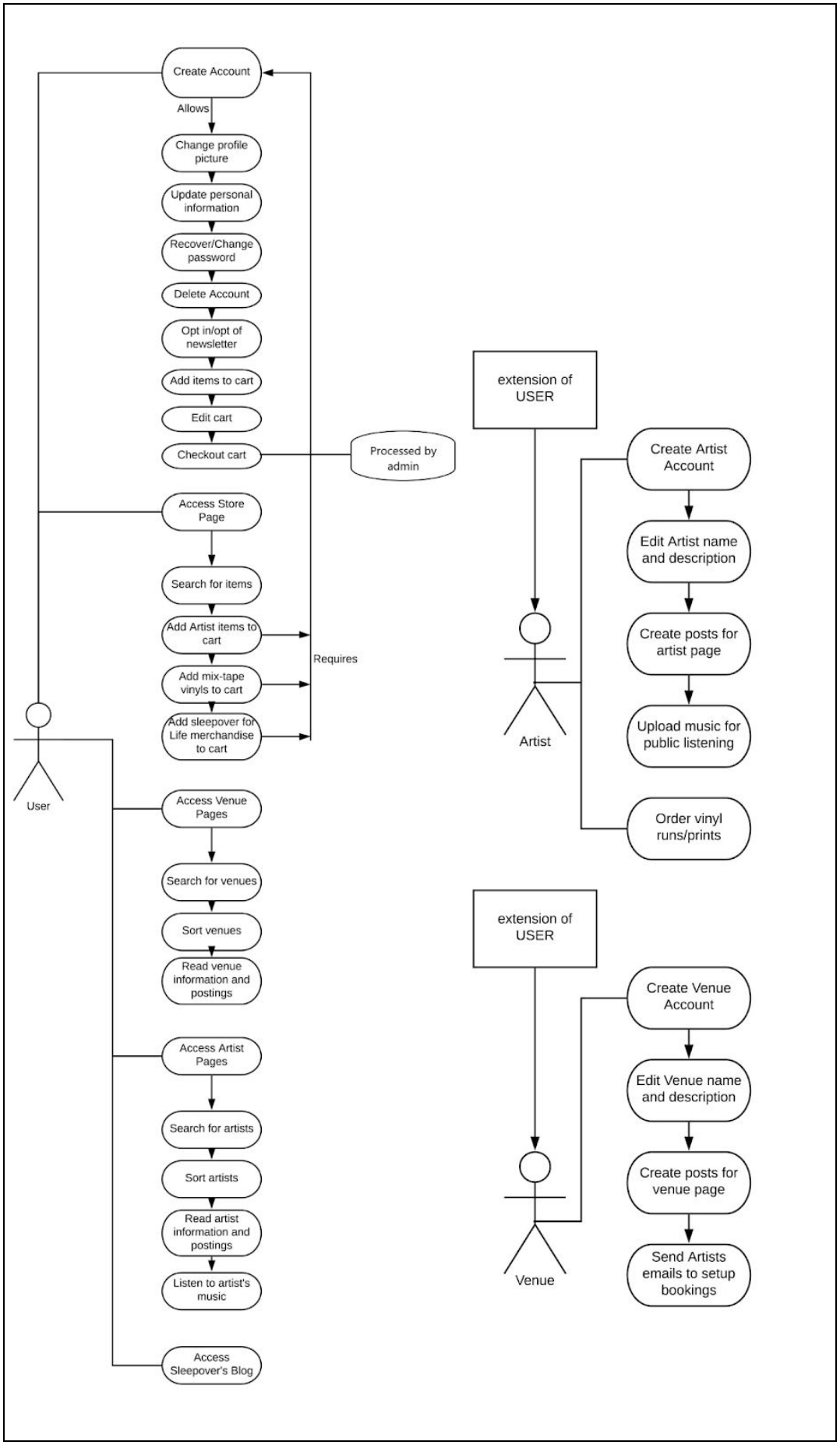
1.0 Project Description

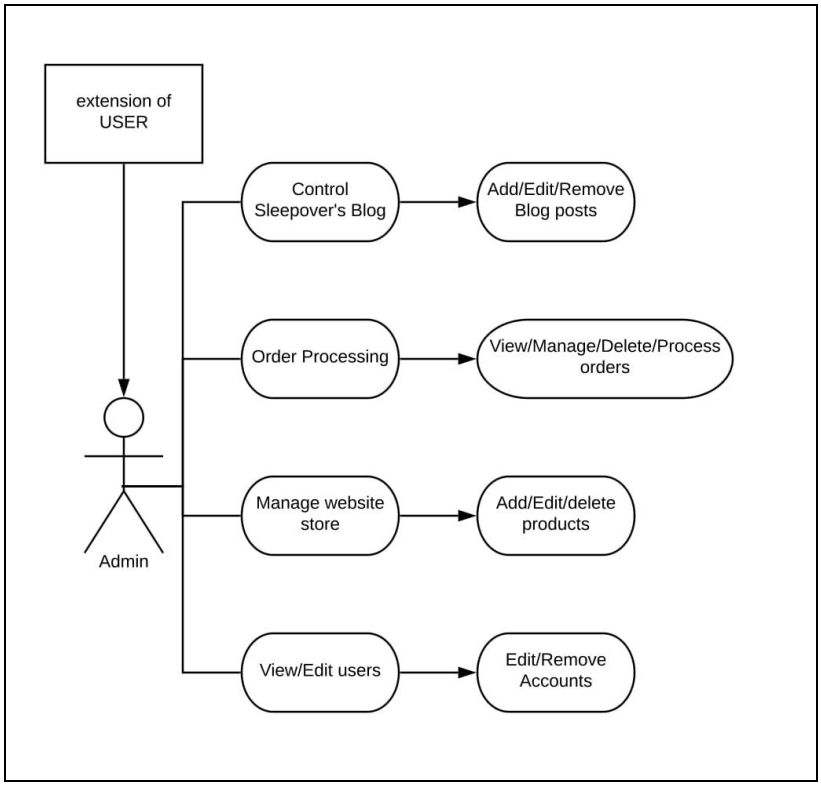
For the Sleepovers project our group is tasked with creating a website for the company, Sleepovers, which will help to provide an affordable way for local musicians in Canada to market themselves and reach out to a larger audience as well as give venues a fast way to get local talent to perform at their location. The artists that sign up for Sleepovers for Life will be given the ability to upload their own music files directly to the website to allow fans to listen without having to leave the site. The website will also let the artists take their original music and purchase a batch of vinyls to sell as physical copies. The other major feature for Sleepovers is that venues will be able to send requests to have bands/artists perform at their location for a given date. Sleepovers will be a website that is used by both the fans, artists, and venues, allowing for all users to create accounts, purchase music and merchandise, and be a part of the Sleepovers blog to keep up to date with the music scene in Canada.

2.0 User Groups

1. Sleepovers Admins / Staff: This will include all workers with Sleepovers that will have control over and manage the website.
2. Artists: Artists will include the bands/musicians that sign up to be a part of Sleepovers.
3. Venues: The venues will be the locations in Kelowna where artists can perform.
4. Public Customers: Sleepovers will be a public website where anyone can become a member to pay more attention to their local music scene.

3.0 Usage Scenarios


Create Account
Allows
Change profile picture
Update personal information
Recover/Change password
Delete Account
Opt in/opt of newsletter
Add items to cart
Edit cart
Checkout cart
Processed by admin
Access Store Page
Search for items
Add Artist items to cart
Add mix-tape vinyls to cart
Add sleepover for Life merchandise to cart
Requires
User
Access Venue Pages
Search for venues
Sort venues
Read venue information and postings
Access Artist Pages
Search for artists
Sort artists
Read artist information and postings
Listen to artist's music
Access Sleepover's Blog
extension of USER
Artist
Create Artist Account
Edit Artist name and description
Create posts for artist page
Upload music for public listening
Order vinyl runs/prints
extension of USER
Venue
Create Venue Account
Edit Venue name and description
Create posts for venue page
Send Artists emails to setup bookings



Figure(s) These figures display the use case scenarios on our site. Each major function can have multiple functions included within. The actor on the left signifies which user we are referring to. The functions attached to the actor lead to other functions that are available to the user via the main functions. Some functions involve other users and are also labelled to the right.

4.0 System Architecture

4.1 Data Flow Diagram

4.1.1 Context Diagram

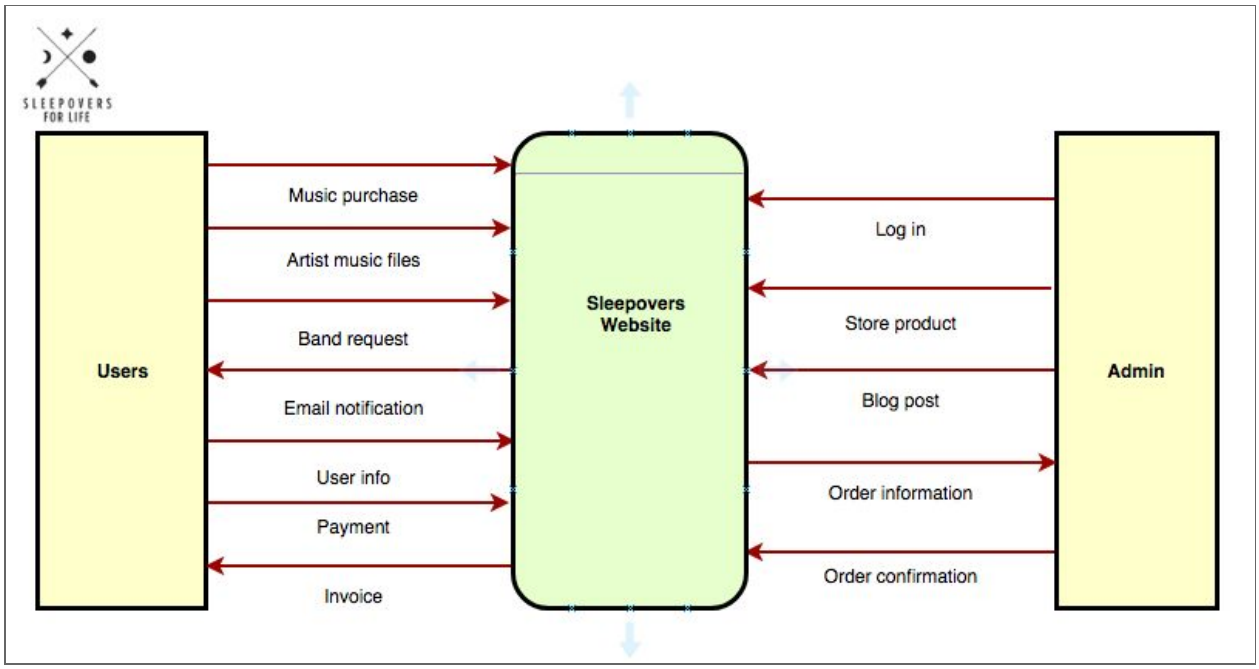


Figure 2.1 The image displays the basic data that will flow into and out of the Sleepovers website. The data that is being sent to/from the user is for the creation of a Sleepovers account, the purchasing of music, requesting a band/artist for a performance and feedback from the website in the form of purchase invoices and email notifications for requests. Admin's will send data to the website to confirm orders, add to the store and post to the blog only receiving information regarding customer purchases.

Main Entities:

1. User: The user is broken down into 3 categories listed below and is meant to represent the primary user group that will be operating on the site. The user will be the primary group to interact with the website and engage in purchasing, browsing and listening to music from the site.
    a. Artist/Band
    b. Venue
    c. Customer
2. Admin: The Admin entity deals with the overall maintenance of the website and database for Sleepovers. Admins will receive information regarding customer orders, and have the power to upload to the store and post to the Sleepovers blog.

4.1.2 Level 1 DFD

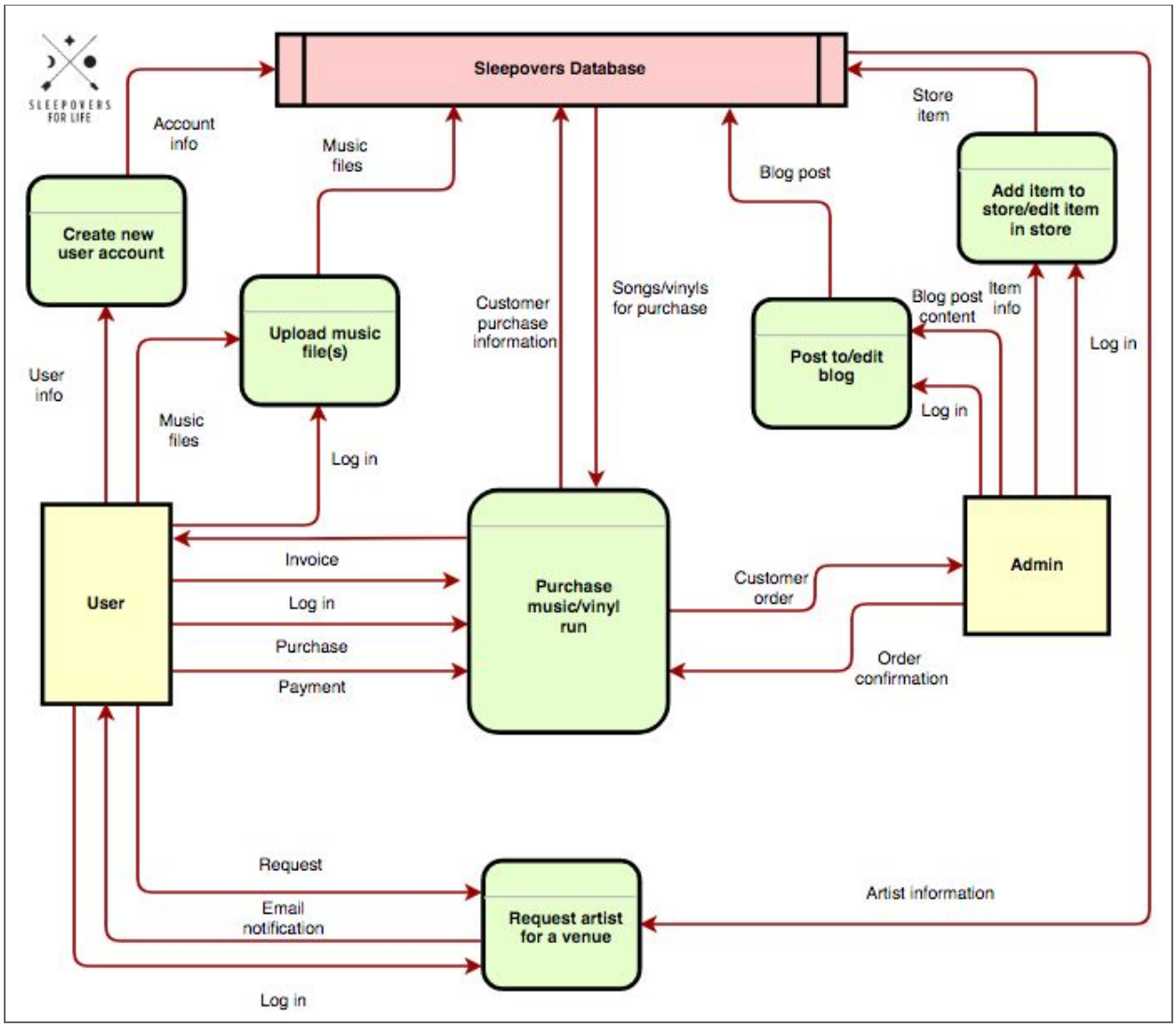


Figure 2.2 is the next level for the Sleepovers website displaying the main processes set up to handle the different data that will be flowing in/out of the website.

Processes:

1. Purchase music/vinyl run: This process will handle the purchase of vinyl records with specific songs. It will check that the user is a valid customer that has an account with Sleepovers, before confirming the order with an admin then sending the user an invoice and product information to the database.
2. Create new user account: The creation of a new user account applies for venues, artists, and fans that needs to first guarantee that the user has a valid email address and password. Before the account is added to the database the user needs to decide whether they would like to opt-in to the mailing list for the Sleepovers blog.

3. Upload music file(s): For the uploading of music process it must check that the artist/band has a valid account with Sleepovers. The artist music also agree to the legal terms before they are allowed to upload any original music.
4. Request artist for a venue: The process that handles an artist/band request takes the request for a band and forms that into an email notification sent to the artist as a way for the venue to get in contact with the artist safely.
5. Post to blog: The “Post to blog” process will create the blog post as an entry into the database for ease of access and storage to allow it to then be posted onto the blog page.
6. Add item to store: The process for adding items to the store works in the same way as the blog post by first creating an item entry for the database that contains the item information that is then directly added to the database to then be shown in the store.

4.1.3 Level 2 DFD

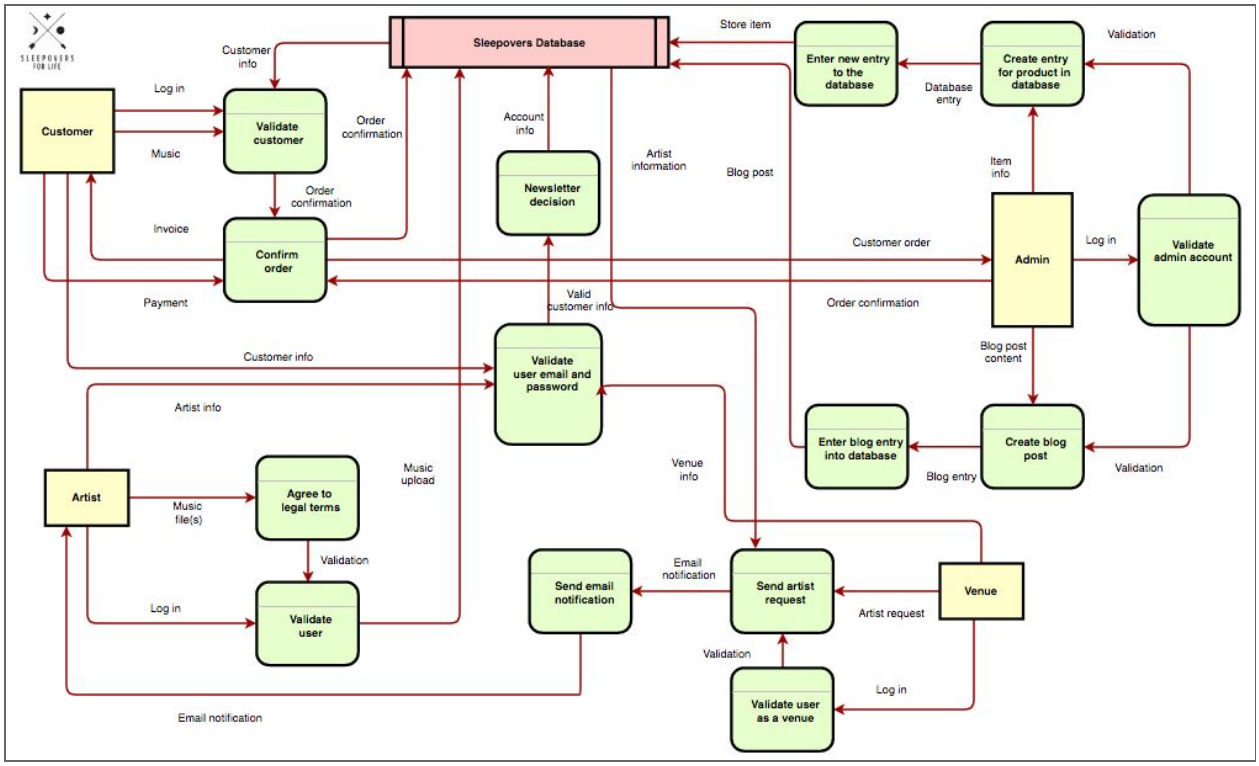


Figure 2.3 shows the expanded view of the level 1 DFD shown above in figure 2.2 that breaks down the User entity into the different entities that can interact individually with the sub-processes for each main process listed in figure 2.2.

The User entity was split into the three different types of users since only certain users are allowed to interact with certain processes such as the requesting of an artist by a venue.

4.2 ER Diagram

[ER Diagram Link]

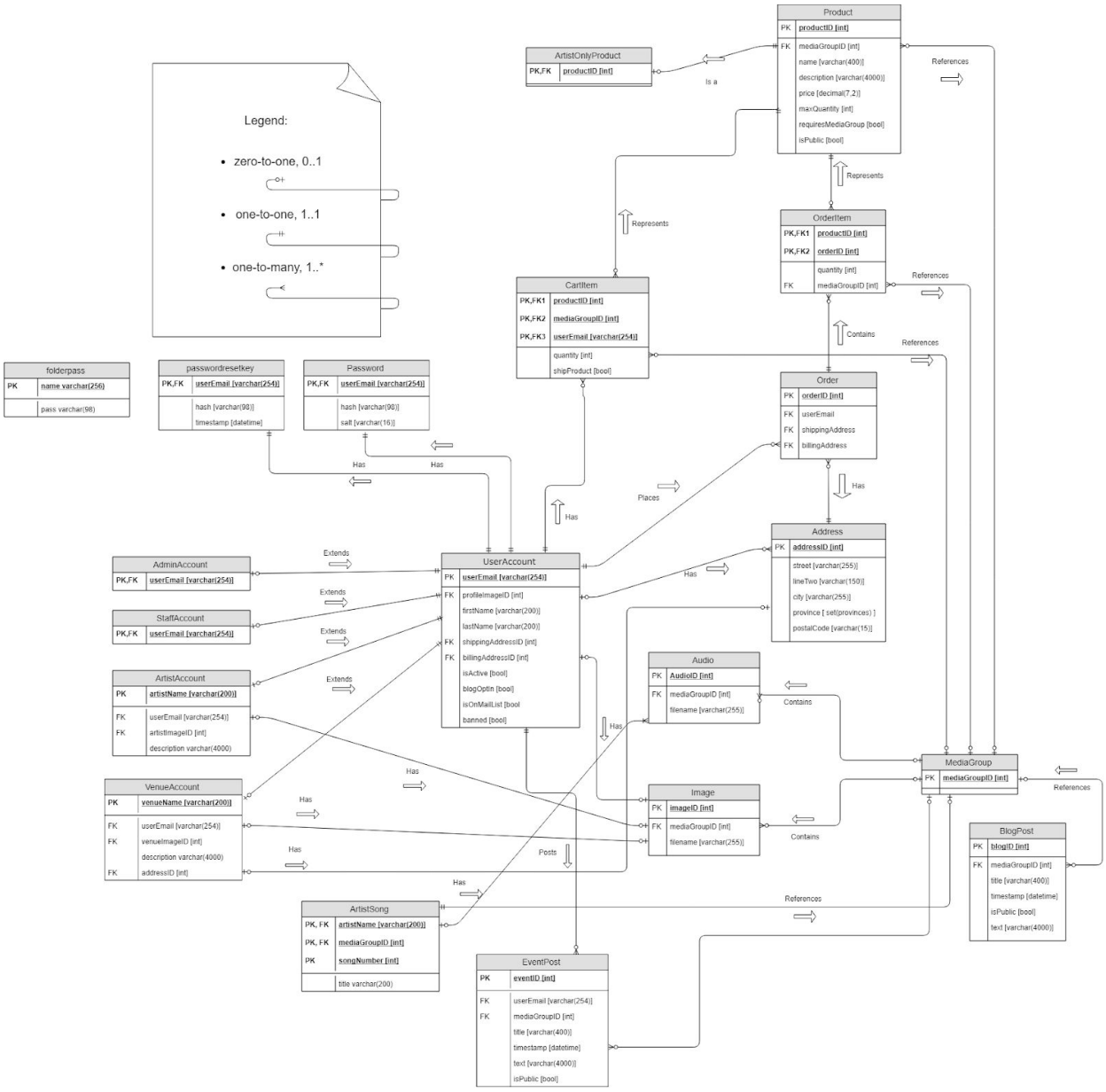


Figure 3.0

The ER diagram for the Sleepovers project models our entire backend database that we will be using to store user and website data. The diagram is mainly centered around the UserAccount table, which contains the attributes associated with all basic accounts and can be extended to be included in the AdminAccount, StaffAccount, ArtistAccount, or VenueAccount tables as well. The basic UserAccount also has the capabilities to post an event to their page or include a profile image. Artist and Venue accounts have the added ability to include a banner

image to showcase their brand. All UserAccount entries are also associated with the Password table which stores a hash value and salt value which will be used to store the hash value output from the Argon2i hash algorithm and the salt that corresponds to it. This allows the website to never store user passwords and makes sure that it conforms with modern standards.

The database will also store user cart items to allow a user to keep their cart contents even if they close the browser. When a user checks out, the corresponding entries in the CartItem table are transferred to the OrderItem table and associated with the appropriate orderID from the Order table. Both the CartItem and OrderItem tables reference products from the Product table. Since some purchases will only be available to artists, the ArtistOnlyProduct table exists to whitelist certain products to the Artists' webstore.

A passwordResetKey table was added to allow temporary hash values to be stored while a user is trying to reset their password. These has a timestamp and will timeout if the user takes too long.

The folderPass table was added to store Trie Search Catalog file paths that contain sensitive information such as user emails. It would be easy for a hacker to find and download the TSC file if it was in plain text, so this table stores a temporary folder name that the file is stored in that is very difficult to guess. This folder changes every time the TSC file is updated (Maximum every 30 seconds). This is the only completely separate table in the database.

Finally, To handle media storage, all of the Entities that require bulk media elements reference a mediaGroupID which can be used to query for all of the image or audio files associated with that ID. Image and Audio tables store the local server path to the file and can be used to retrieve uploaded files. Currently, .txt files are a rarety to be uploaded to the server, but are sometimes required for custom product media uploads. These are currently stored in the image table as well, but should not affect any functionality since all files are queried by mediaGroupID and the tables are only referenced to separate audio and image files.

4.2.1 ER Schema

```
CREATE TABLE IF NOT EXISTS `passwordresetkey` (
 `userEmail` varchar(254) NOT NULL,
 hash varchar(98) NOT NULL,
 timestamp datetime DEFAULT NULL,
 PRIMARY KEY (`userEmail`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE IF NOT EXISTS `mediagroup` (
 `mediaGroupID` int(11) NOT NULL AUTO_INCREMENT,
 PRIMARY KEY (`mediaGroupID`)
```

```
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE IF NOT EXISTS `audio` (
 `audioID` int(11) NOT NULL AUTO_INCREMENT,
 `mediaGroupID` int(11) DEFAULT NULL,
 `filename` varchar(255) NOT NULL,
 PRIMARY KEY (`audioID`),
 FOREIGN KEY (`mediaGroupID`) REFERENCES mediagroup(`mediaGroupID`)
   ON DELETE SET NULL ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE IF NOT EXISTS `blogpost` (
 `blogID` int(11) NOT NULL AUTO_INCREMENT,
 `mediaGroupID` int(11) DEFAULT NULL,
 `title` varchar(400) NOT NULL,
 `timestamp` datetime DEFAULT NULL,
 `isPublic` BOOLEAN DEFAULT 0,
 `text` varchar(4000) DEFAULT NULL,
 PRIMARY KEY (`blogID`),
 FOREIGN KEY (`mediaGroupID`) REFERENCES mediagroup(`mediaGroupID`)
   ON DELETE SET NULL ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE IF NOT EXISTS `image` (
 `imageID` int(11) NOT NULL AUTO_INCREMENT,
 `mediaGroupID` int(11) DEFAULT NULL,
 `filename` varchar(255) NOT NULL,
 PRIMARY KEY (`imageID`),
 FOREIGN KEY (`mediaGroupID`) REFERENCES mediagroup(`mediaGroupID`)
   ON DELETE SET NULL ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE IF NOT EXISTS `product` (
 `productID` int(11) NOT NULL AUTO_INCREMENT,
 `mediaGroupID` int(11) DEFAULT NULL,
 `name` varchar(400) NOT NULL,
 `description` varchar(4000) DEFAULT NULL,
```

```
 `price` decimal(7,2) NOT NULL,
 `maxQuantity` int(11) DEFAULT NULL,
 `requiresMediaGroup` BOOLEAN DEFAULT 0,
 `isPublic` BOOLEAN DEFAULT 0,
 PRIMARY KEY (`productID`),
 FOREIGN KEY (`mediaGroupID`) REFERENCES mediagroup(`mediaGroupID`)
   ON DELETE SET NULL ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=latin1;


CREATE TABLE IF NOT EXISTS `artistonlyproduct` (
 `productID` int(11) NOT NULL,
 PRIMARY KEY (`productID`),
 FOREIGN KEY (`productID`) REFERENCES product(`productID`)
   ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE IF NOT EXISTS `address` (
 `AddressID` int(11) NOT NULL AUTO_INCREMENT,
 `street` varchar(255) NOT NULL,
 `lineTwo` varchar(150) DEFAULT NULL,
 `city` varchar(255) NOT NULL,
 `province` SET('BC','AB','MB','NB','NL','NT','NS','NU','ON','PE','QC','SK','YT')
           NOT NULL,
 `postalCode` varchar(15) NOT NULL,
 PRIMARY KEY (`AddressID`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE IF NOT EXISTS `useraccount` (
 `userEmail` varchar(254) NOT NULL,
 `profileImageID` int(11) DEFAULT NULL,
 `firstName` varchar(200) NOT NULL,
 `lastName` varchar(200) NOT NULL,
 `shippingAddressID` int(11) DEFAULT NULL,
 `billingAddressID` int(11) DEFAULT NULL,
 `isActive` BOOLEAN DEFAULT 1,
 `blogOptIn` BOOLEAN DEFAULT 0,
```

```
 `isOnMailList` BOOLEAN DEFAULT 0,
 `banned` BOOLEAN DEFAULT 0,
 PRIMARY KEY (`userEmail`),
 FOREIGN KEY (`profileImageID`) REFERENCES image(`imageID`)
   ON DELETE SET NULL ON UPDATE CASCADE,
 FOREIGN KEY (`shippingAddressID`) REFERENCES address(`addressID`)
   ON DELETE SET NULL ON UPDATE CASCADE,
 FOREIGN KEY (`billingAddressID`) REFERENCES address(`addressID`)
   ON DELETE SET NULL ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE IF NOT EXISTS `eventpost` (
 `eventID` int(11) NOT NULL AUTO_INCREMENT,
 `userEmail` varchar(254) NOT NULL,
 `mediaGroupID` int(11) DEFAULT NULL,
 `title` varchar(400) NOT NULL,
 `timestamp` datetime DEFAULT NULL,
 `text` varchar(4000) DEFAULT NULL,
 `isPublic` BOOLEAN DEFAULT 0,
 PRIMARY KEY (`eventID`),
 FOREIGN KEY (`mediaGroupID`) REFERENCES mediagroup(`mediaGroupID`)
   ON DELETE SET NULL ON UPDATE CASCADE,
 FOREIGN KEY (`userEmail`) REFERENCES useraccount(`userEmail`)
   ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE IF NOT EXISTS `adminaccount` (
 `userEmail` varchar(254) NOT NULL,
 PRIMARY KEY (`userEmail`),
 FOREIGN KEY (`userEmail`) REFERENCES useraccount(`userEmail`)
   ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE IF NOT EXISTS `artistaccount` (
 `artistName` varchar(200) NOT NULL,
 `userEmail` varchar(254) NOT NULL,
 `artistImageID` int(11) DEFAULT NULL,
```

```
 `description` varchar(4000) DEFAULT NULL,
 PRIMARY KEY (`artistName`),
 FOREIGN KEY (`userEmail`) REFERENCES useraccount(`userEmail`)
   ON DELETE CASCADE ON UPDATE CASCADE,
 FOREIGN KEY (`artistImageID`) REFERENCES image(`imageID`)
   ON DELETE SET NULL ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE IF NOT EXISTS `cartitem` (
 `productID` int(11) NOT NULL,
 `userEmail` varchar(254) NOT NULL,
 `mediaGroupID` int(11) NOT NULL,
 `quantity` int(11) DEFAULT 1,
 `shipProduct` BOOLEAN DEFAULT 1,
 PRIMARY KEY (`productID`,`userEmail`, `mediaGroupID`),
 FOREIGN KEY (`productID`) REFERENCES product(`productID`)
   ON DELETE CASCADE ON UPDATE CASCADE,
 FOREIGN KEY (`userEmail`) REFERENCES useraccount(`userEmail`)
   ON DELETE CASCADE ON UPDATE CASCADE,
 FOREIGN KEY (`mediaGroupID`) REFERENCES mediagroup(`mediaGroupID`)
   ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE IF NOT EXISTS `orders` (
 `orderID` int(11) NOT NULL AUTO_INCREMENT,
 `userEmail` varchar(254) NOT NULL,
 `shippingAddress` int(11) NOT NULL,
 `billingAddress` int(11) NOT NULL,
 `orderStatus` SET('uncon','con','comp','ship') DEFAULT 'uncon',
 `orderDate` datetime DEFAULT NULL,
 `completionDate` datetime DEFAULT NULL,
 `shipDate` datetime DEFAULT NULL,
 PRIMARY KEY (`orderID`),
 FOREIGN KEY (`userEmail`) REFERENCES useraccount(`userEmail`)
   ON DELETE NO ACTION ON UPDATE CASCADE,
 FOREIGN KEY (`shippingAddress`) REFERENCES address(`addressID`)
   ON DELETE NO ACTION ON UPDATE CASCADE,
```

```
 FOREIGN KEY (`billingAddress`) REFERENCES address(`addressID`)
   ON DELETE NO ACTION ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE IF NOT EXISTS `orderitem` (
 `productID` int(11) NOT NULL,
 `orderID` int(11) NOT NULL,
 `quantity` varchar(255) NOT NULL,
 `mediaGroupID` int(11) NOT NULL,
 `shipProduct` BOOLEAN DEFAULT 1,
 `orderStatus` SET('uncomp','comp') DEFAULT 'uncomp',
 PRIMARY KEY (`productID`,`orderID`,`mediaGroupID`),
 FOREIGN KEY (`productID`) REFERENCES product(`productID`)
   ON DELETE CASCADE ON UPDATE CASCADE,
 FOREIGN KEY (`orderID`) REFERENCES orders(`orderID`)
   ON DELETE CASCADE ON UPDATE CASCADE,
 FOREIGN KEY (`mediaGroupID`) REFERENCES mediagroup(`mediaGroupID`)
   ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE IF NOT EXISTS `password` (
 `userEmail` varchar(254) NOT NULL,
 `hash` varchar(98) NOT NULL,
 PRIMARY KEY (`userEmail`),
 FOREIGN KEY (`userEmail`) REFERENCES useraccount(`userEmail`)
   ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE IF NOT EXISTS `staffaccount` (
 `userEmail` varchar(254) NOT NULL,
 PRIMARY KEY (`userEmail`),
 FOREIGN KEY (`userEmail`) REFERENCES useraccount(`userEmail`)
   ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE IF NOT EXISTS `venueaccount` (
 `venueName` varchar(200) NOT NULL,
```

```
 `userEmail` varchar(254) NOT NULL,
 `venueImageID` int(11) DEFAULT NULL,
 `description` varchar(4000) DEFAULT NULL,
 `addressID` int(11) DEFAULT NULL,
 PRIMARY KEY (`venueName`),
 FOREIGN KEY (`userEmail`) REFERENCES useraccount(`userEmail`)
   ON DELETE CASCADE ON UPDATE CASCADE,
 FOREIGN KEY (`venueImageID`) REFERENCES image(`imageID`)
   ON DELETE SET NULL ON UPDATE CASCADE,
 FOREIGN KEY (`addressID`) REFERENCES address(`AddressID`)
   ON DELETE SET NULL ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE IF NOT EXISTS `artistsong` (
 `artistName` varchar(200) NOT NULL,
 `mediaGroupID` int(11) NOT NULL,
 `songNumber` int(11) NOT NULL,
 `title` varchar(200) DEFAULT NULL,
 PRIMARY KEY (`artistName`,`mediaGroupID`,`songNumber`),
 FOREIGN KEY (`artistName`) REFERENCES artistaccount(`artistName`)
   ON DELETE CASCADE ON UPDATE CASCADE,
 FOREIGN KEY (`mediaGroupID`) REFERENCES mediagroup(`mediaGroupID`)
   ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE IF NOT EXISTS `folderpass` (
 `name` varchar(256) NOT NULL,
 `pass` varchar(98) NOT NULL,
 PRIMARY KEY (`name`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

4.3 Sequence Diagram

4.3.1 Store Purchase

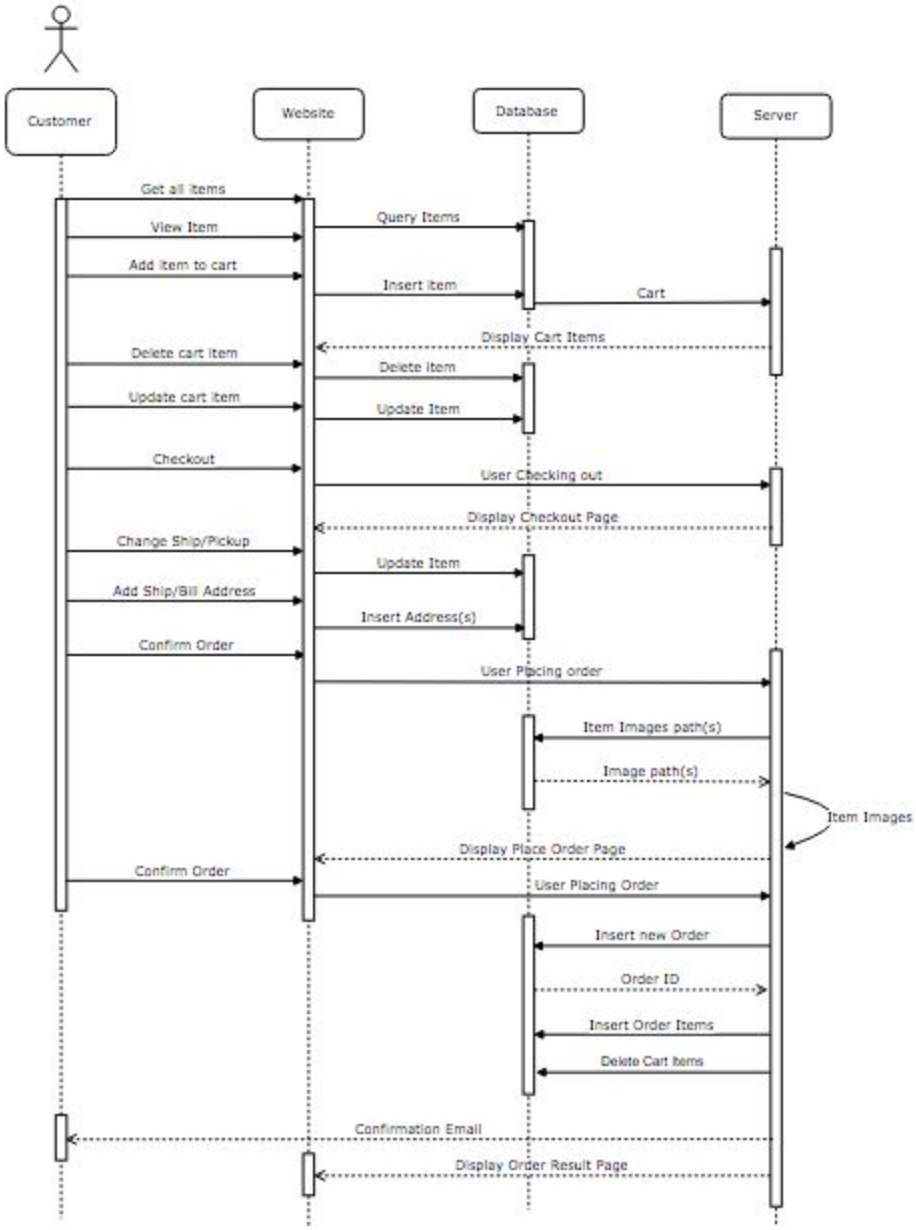


Figure 4.0

The above image lists the steps that a user will most likely take when purchasing an item from the store. The diagram details all the steps that a user/artist can encounter when purchasing an item from the store. The user can add, delete or update items in their cart, change their shipping and billing information for this order, confirm their order and finally place their order.

4.3.2 Customer Mixtape/Vinyl Run

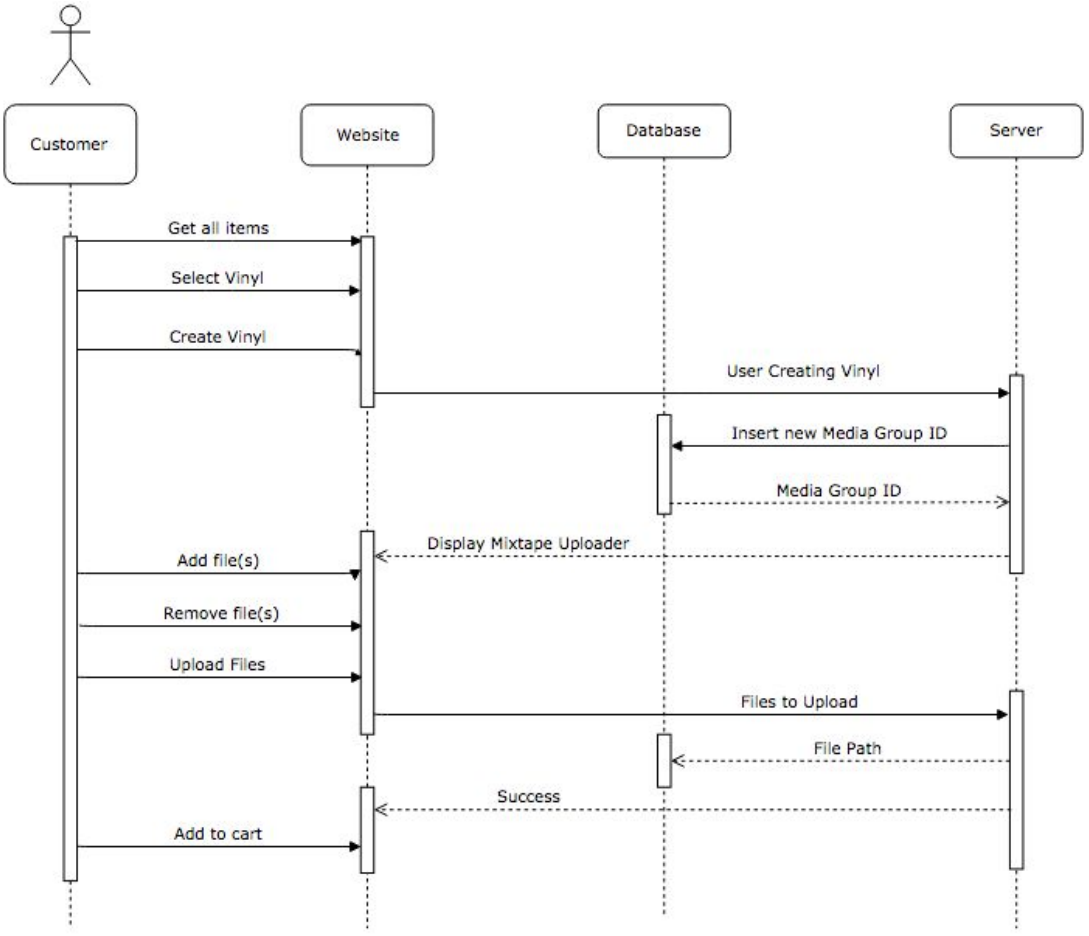


Figure 4.1

The above diagram describes the process that a user/artist would go through to purchase a mixtape or vinyl run. The user will choose the vinyl that they want and can then add their different files for the order which will then be processed by a Sleepovers staff member. This step can also be considered an intermediate step for a purchase of a vinyl.

4.3.3 Create Account/Log in

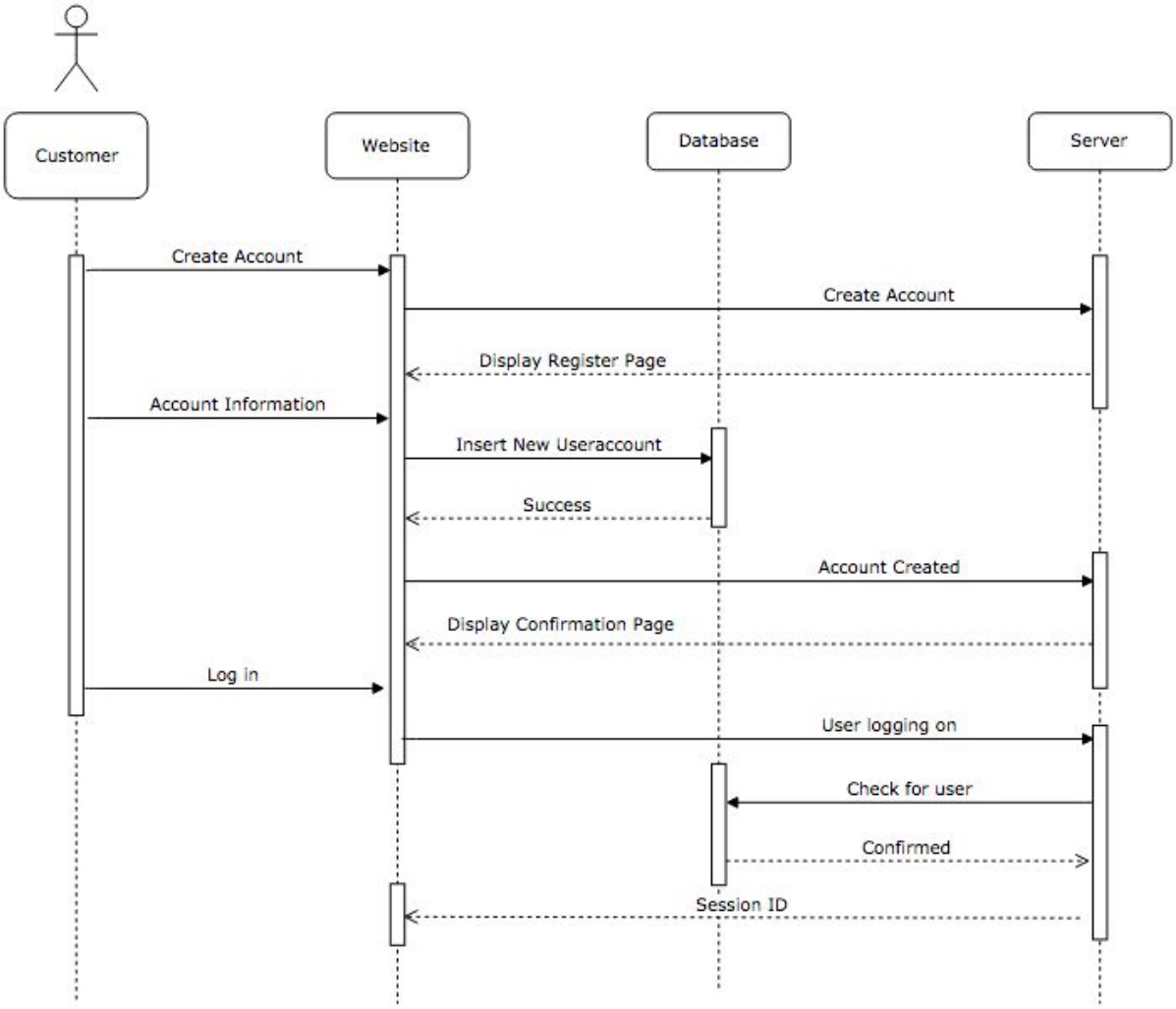


Figure 4.2

The above figure details the process that a user (artist, venue, public customer) would go through in order to create an account and log in using that account. The account information will include email, first and last name, password, what type of user they are (artist, venue, customer) and whether they would like to sign up for notifications on the blog and newsletter provided by Sleepovers.

4.3.4 Artist/Venue/Item/Search

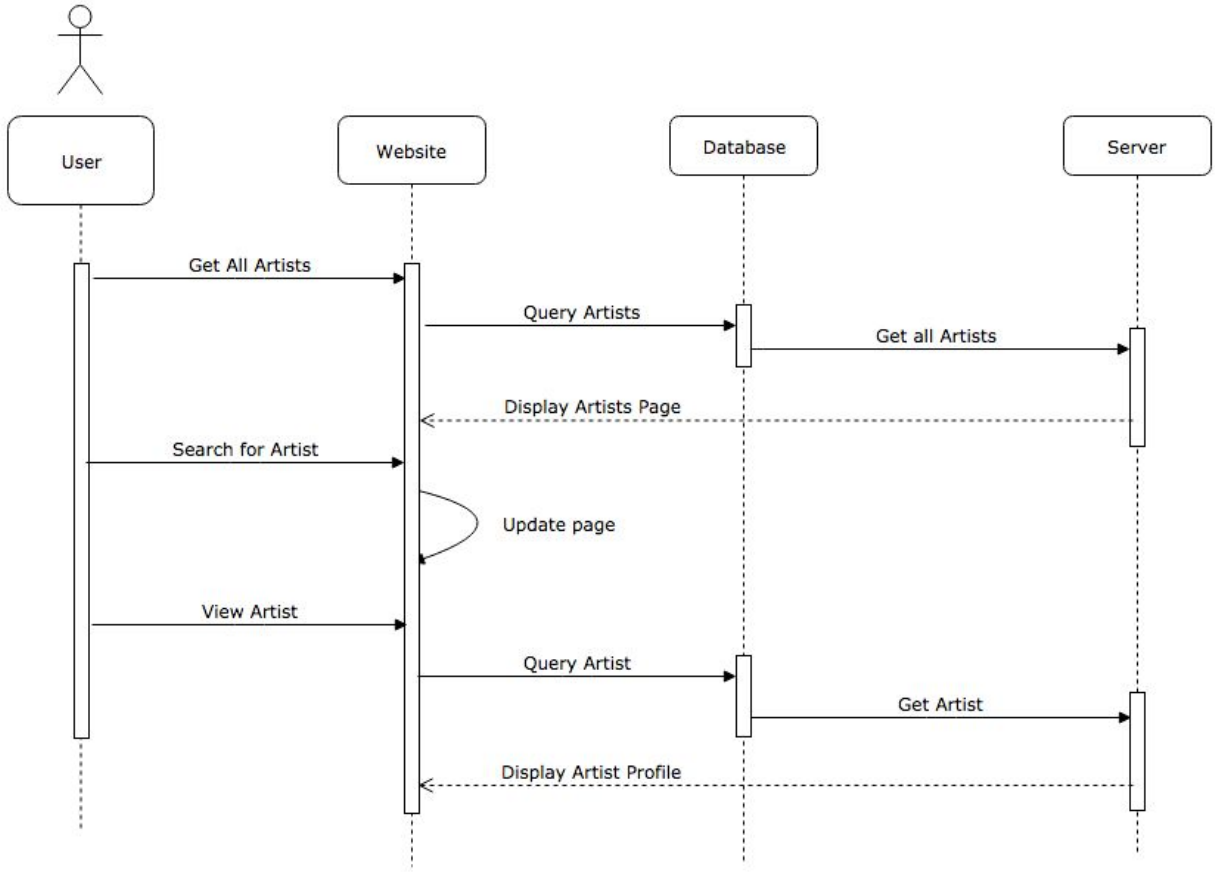


Figure 4.4

The above diagram details the process that a user would go through in order to search for a particular artist/venue/item on the website. The process is the same for each of the items being searched.

4.3.5 Admin Create Product/Blog Post

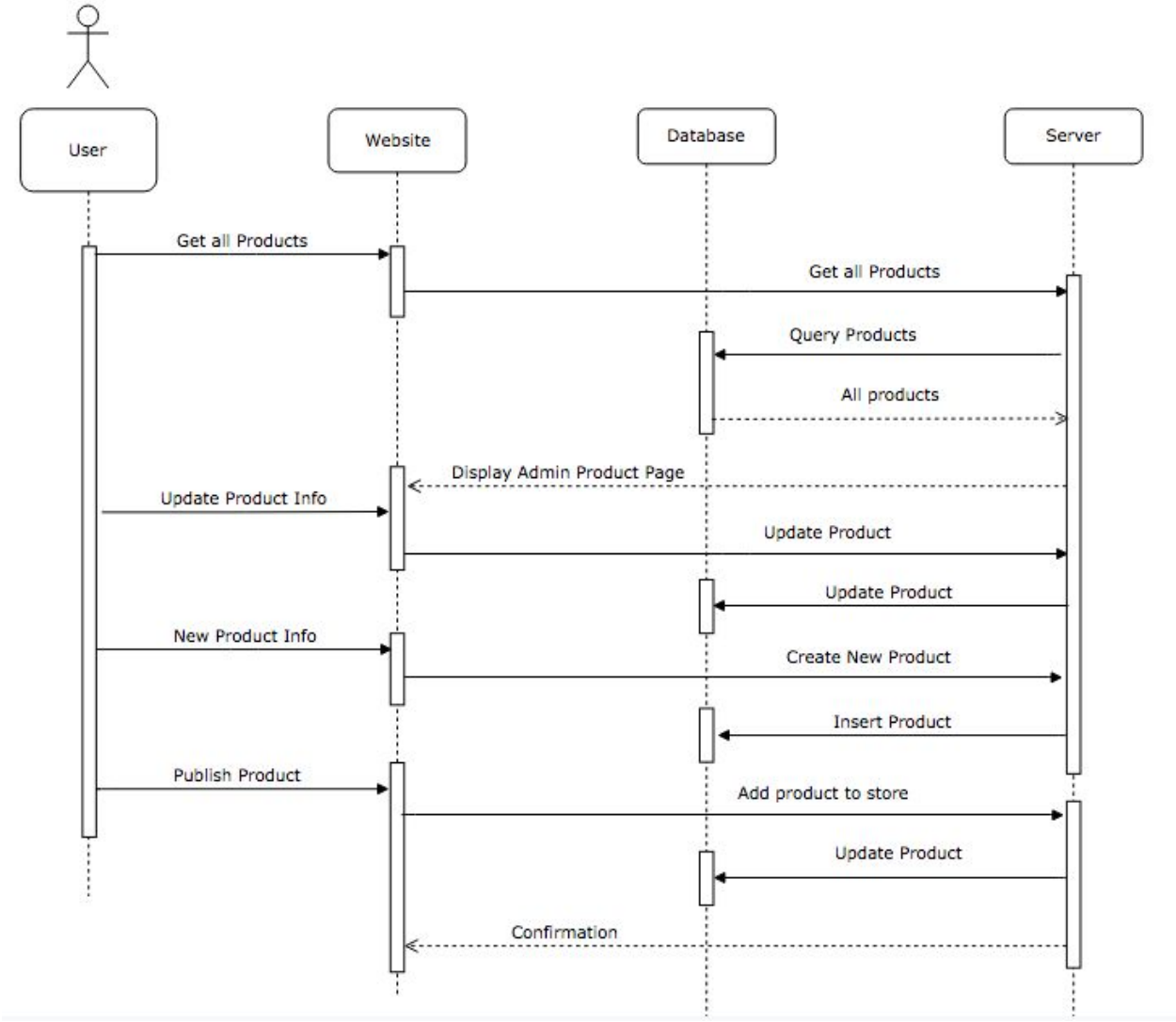
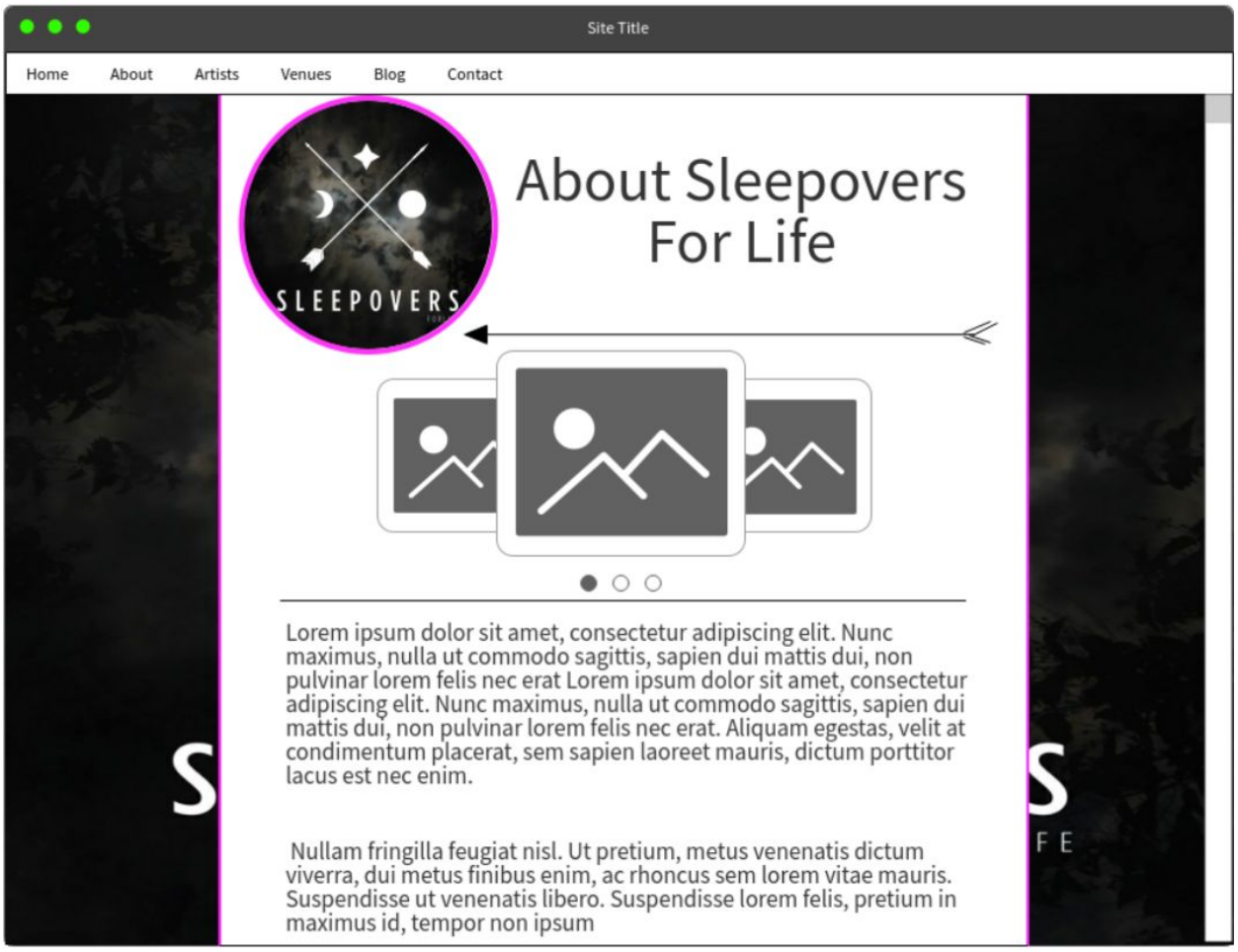


Figure 4.5

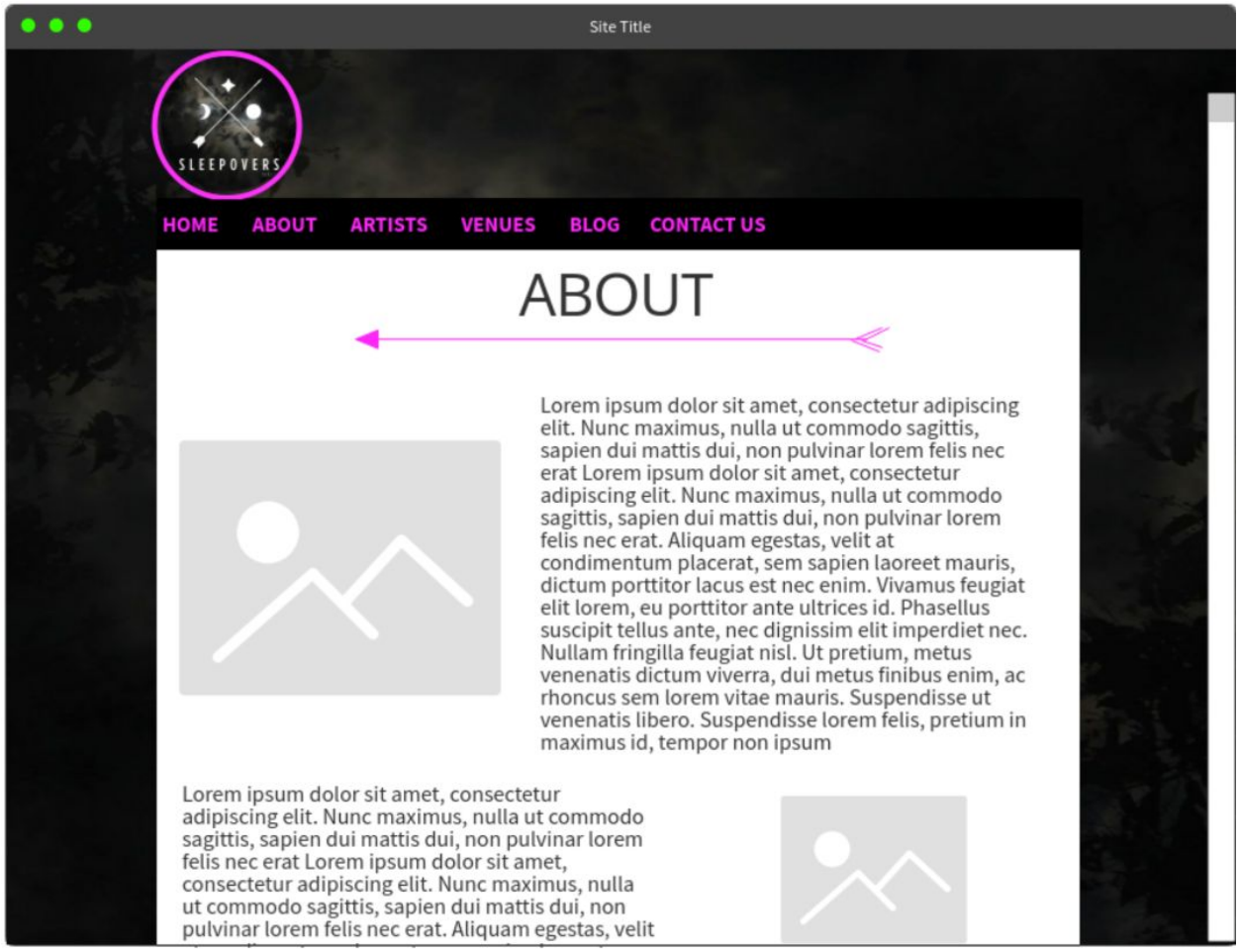
The above diagram details the process that an admin/staff member would go through in order to create a new product/blog post and add it to the website.

5.0 UI Mockups

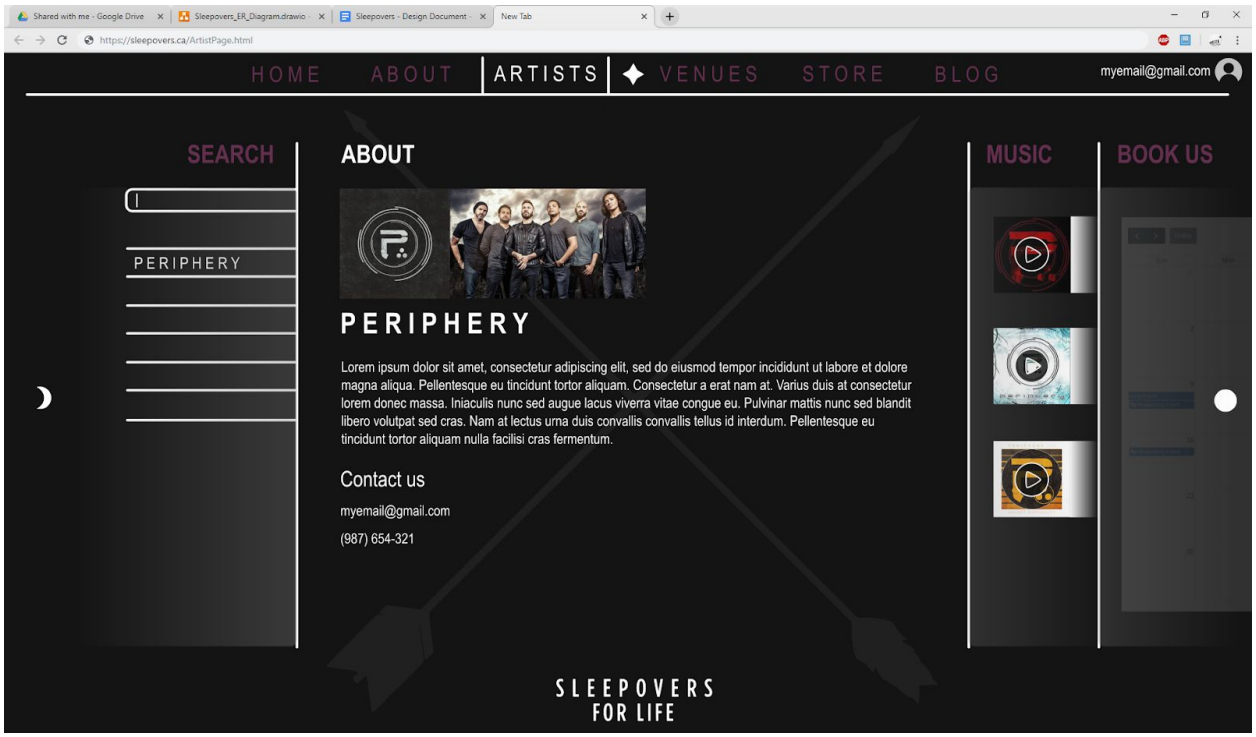
5.1 Example 1



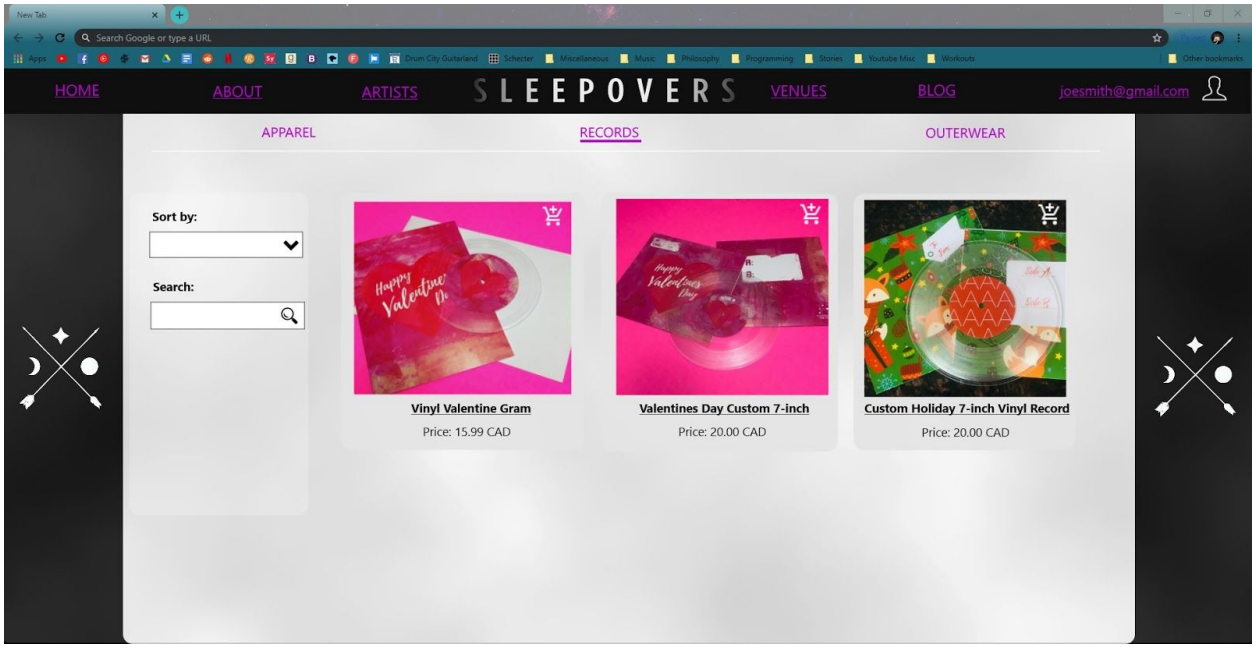
5.2 Example 2


Site Title
SLEEPOVERS
HOME
ABOUT
ARTISTS
VENUES
BLOG
CONTACT US
ABOUT
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc maximus, nulla ut commodo sagittis, sapien dui mattis dui, non pulvinar lorem felis nec erat Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc maximus, nulla ut commodo sagittis, sapien dui mattis dui, non pulvinar lorem felis nec erat. Aliquam egestas, velit at condimentum placerat, sem sapien laoreet mauris, dictum porttitor lacus est nec enim. Vivamus feugiat elit lorem, eu porttitor ante ultrices id. Phasellus suscipit tellus ante, nec dignissim elit imperdiet nec. Nullam fringilla feugiat nisl. Ut pretium, metus venenatis dictum viverra, dui metus finibus enim, ac rhoncus sem lorem vitae mauris. Suspendisse ut venenatis libero. Suspendisse lorem felis, pretium in maximus id, tempor non ipsum
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc maximus, nulla ut commodo sagittis, sapien dui mattis dui, non pulvinar lorem felis nec erat Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc maximus, nulla ut commodo sagittis, sapien dui mattis dui, non pulvinar lorem felis nec erat. Aliquam egestas, velit

5.3 Example 3



5.4 Example 4



5.5 Client Feedback

Based on the mockups sent, the client gave us feedback to consider while completing the project. Their comments included:

- Specific colours to be used (#FF0080)
- The logo design and layout of example 4 was a leading favourite.
- The organization and theme of example 3 was also appreciated

Overall the ideas in examples 3 and 4 were chosen as favourites, but the client allowed some freedom of design saying, “let’s see what it’s like when they follow through with their ideas”.

6.0 Technical Specifications

6.1 MySQL Database

The Sleepovers website will be built using a MySQL database to store all information used on the website excluding audio and image files. The database will store user profiles, webstore products, blog posts and any other user-manipulated data with the goal to keep all of the information accessible to the public and easy to manipulate by the author and Sleepovers’ staff. Audio (.wav, .flac, .mp3) and image files (.jpg, .jpeg, .png, .bmp, .gif) will be stored on the server with the file paths being stored in the database for each file. The website will also work with the database to bring a simple order management system for Sleepovers staff to easily view and complete customer orders. We will use PHP as our server side language in order to perform queries to and from the database using the website.

6.2 Server

Both the website and the database are to be developed using the COSC 499 server provided by Scott Fazackerley and will support running PHP 7 and higher. The website will be developed using the most recent version of PHP 7.3.6 in order to guarantee the functionality of the ARGON 2i password hashing in order to protect the users information.

6.3 Plupload

In order to allow users to upload audio and image files for use on their profiles or for purchasing records, the Sleepovers website will make use of an upload software called Plupload which will break the file down into chunks and store the files on the server. Plupload allows us to set variables such as chunk size, maximum allowed file size, and what types of files are accepted. We can create custom instances of the uploader interface depending on our needs. Multiple file uploads are also controlled by the uploader and all progress can be tracked

visually. For instance, profile pictures can be uploaded and will instantly be applied as the new profile picture for the active user. We are able to do so without redirecting or refreshing any pages, which allows the process to be user friendly and technically simple.

6.4 PHPMailer

The Sleepovers website will need to be able to send emails to confirm orders, and perform password resets and in order to do that we will be setting up a mail service for the website using PHPMailer. PHPMailer is an open source software that works with PHP to send emails and does not require a mail server to run. PHPMailer is setup to send emails using SMTP and is encrypted using TLS (an upgraded version of SSL), can be used to send to multiple To, BCC, CC and Reply-to address, and can send HTML emails with multiple attachments including files and images.

6.5 Website Design

The Sleepovers website will be coded in HTML5, CSS, and Javascript (ECMAScript 3) which will allow our team to implement interactive elements that are required for the project. Bootstrap v4.3.1 will be used for front end development to make the website usable on both larger laptop style screens as well as smaller mobile screens.

7.0 Testing Plan

1. Query testing for the database
    a. As we initialize the database and every time we make changes to the design in the future, we will create tables and sample entities and test that correct query results are returned. An example of this would be to add users to the UserAccount table and query for particular accounts similar to how an admin or Sleepovers staff member would if looking for accounts.
2. Integration testing with PHP and SQL
    a. As we start to develop the PHP backbone of our website and integrate the SQL queries into our PHP, we will check that the correct items are displayed on every page before furthering development. This will be primarily tested when we are creating the webstore to ensure that items in the store are being correctly displayed and that all functionality works for adding items to a cart and checking out.
3. Continuous integration testing with TravisCI
    a. Our team will utilize TravisCI to perform continuous integration testing on our PHP files when we merge GitHub branches.
4. Unit testing for javascript functions
    a. As we develop javascript for various website functionality, we will code unit tests for each function and run the script manually on every build to view the test results in the web console. An example of this would be when a user is creating

an account to ensure that the section for the creation of a password to ensure that the password created is within parameters (eg. length, at least 1 character and 1 number).

5. User functionality testing and feedback
    a. As the development reaches the end of the project, we will organize hands-on user meetings with venue managers, local artists, and prospective customers to test user functionality on the website and gather feedback. For these tests we will be in contact with users that our client is aware of to use the site to perform tasks such as uploading songs or requesting an artist for a venue.
6. Acceptance Testing
    a. Near the end of our project, we will ensure that the final deliverable is acceptable to our client and ensure that we are not missing any expected functionality or design. An example of this would be to have our client, Angie, to perform the major tasks on the website such as validating the entire process for vinyl run purchase where she will make a mock-up purchase that will will then be able to confirm.

8.0 Approvals

Project Sponsor Signature: _____ Date: _____

Everton Smith Signature: _____ Date: _____

Joshua Henderson Signature: _____ Date: _____

Trevor Richard Signature: _____ Date: _____