Break The Earth Web Quest

Hiram Viezca & Shawn Fradet

CST-452 Capstone Project Final Architecture & Design

Grand Canyon University

Instructor: Prof. Amr Elchouemi

Revision: 2.0

Date: 4/13/2022

**ABSTRACT**

This design document covers a web application project for a local heavy metal band “Break the Earth”. This band is looking to increase their web presence while centralizing access to existing online media. The band requires a web application that will help them connect with their fans and music industry professionals by providing information about the band and their music. The proposed web application provides information on the band members, their available discography, material in work, live events, and YouTube videos. The web application will also provide the ability to book them for a show and purchase their merchandise.

This design document breaks down all aspects of the web application along with risk mitigation and a schedule for accomplishing everything in our allotted time frame.

|  |
| --- |
| History and Signoff Sheet |

**Change Record**

|  |  |  |
| --- | --- | --- |
| **Date** | **Author** | **Revision Notes** |
| 3/15/2022 | Hiram Viezca & Shawn Fradet | Initial draft for review/discussion |
| 4/11/22 | Hiram Viezca & Shawn Fradet | Final Draft  Updated Album Management to reflect design changes with Track Progress. |
|  |  |  |

|  |
| --- |
| **Overall Instructor Feedback/Comments** |

|  |
| --- |
| **Overall Instructor Feedback/Comments** |

**Integrated Instructor Feedback into Project Documentation**

☐ Yes ☐ No

**TABLE OF CONTENTS**

**Design Overview 4**

**Detailed High-Level Solution Design 5**

**Detailed Technical Design 6**

**Appendix A – Technical Issue and Risk Log 7**

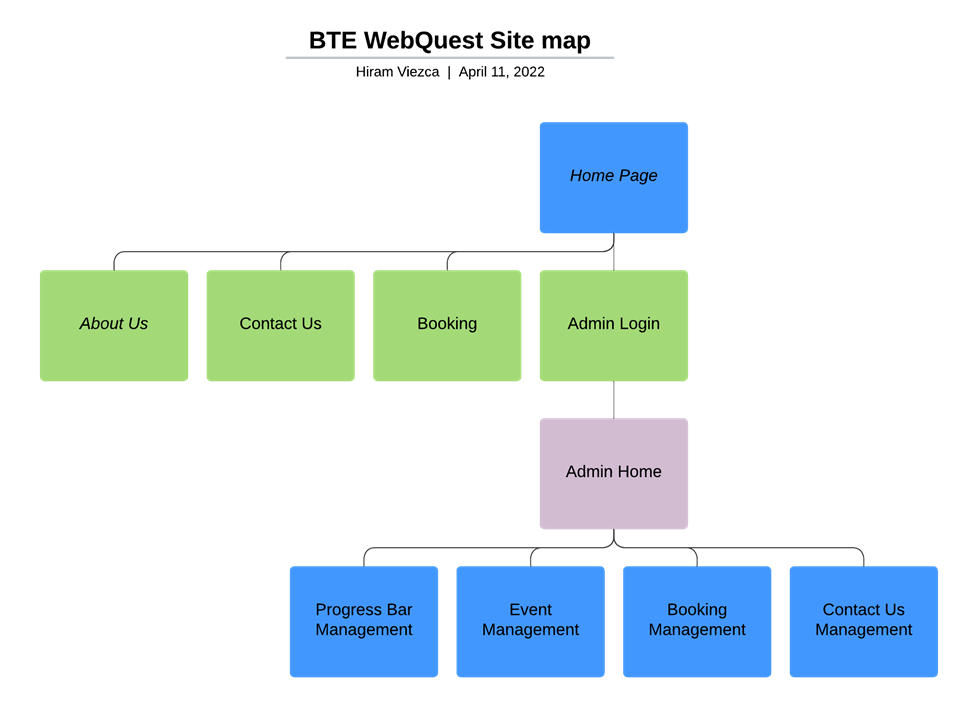
**Appendix B – References 8**

**Appendix C – External Resources 9**

**Design Introduction**

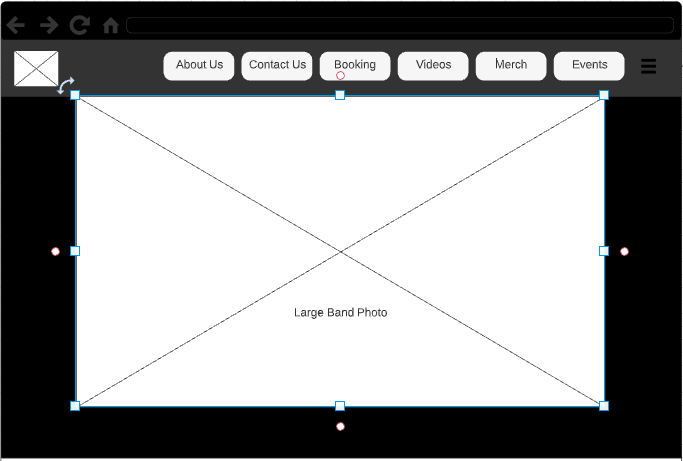
The client for this project is a heavy metal band composed of four members. The objective is to create a dynamic website providing a place for fans to connect, a location for individuals and booking agents to book the band allowing for the increased exposure of the group’s music, and access to the client’s merchandise and music providing a source of revenue. The following is a site map for the website.

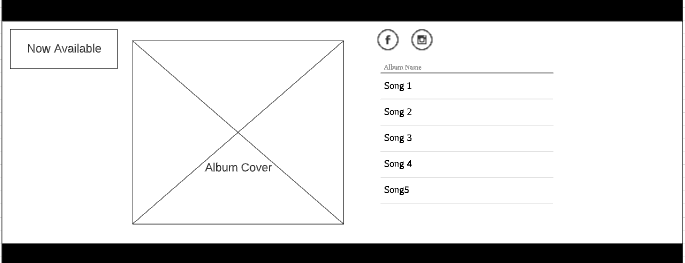
**Sitemap Diagram:**

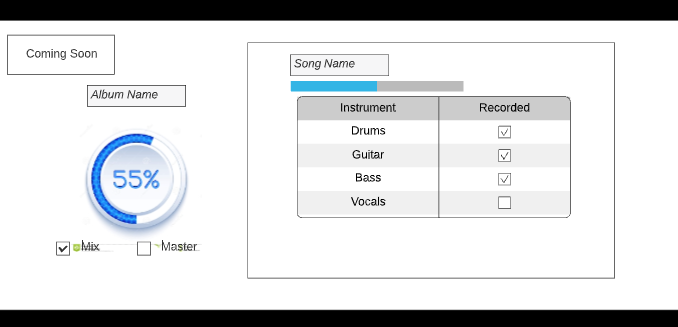


The goal is to produce a modern design for the site so that the user can find the information they seek with as few clicks as possible. The website needs to include an image of the band and should, at minimum, include the following areas:

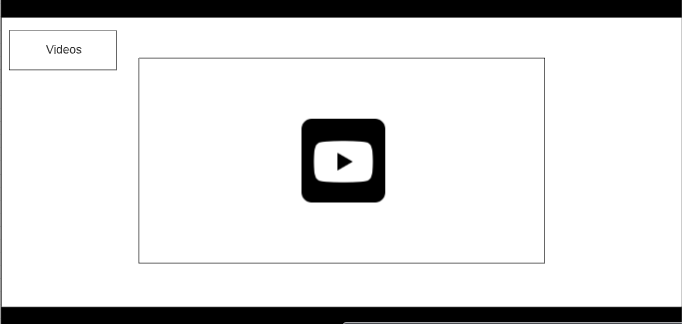
* “Main” page - This page will be the landing page for the website and will provide most of the information that users will usually want to see immediately. The header will have clickable links to the different sections on this page and the page will move itself to the section corresponding to the link clicked. These different sections will be served by Thymeleaf fragments and will include the following:
  + Band image - This section will be an image of the band. It will cover the whole screen and the user will scroll past to the rest of the sections. The Administrator and site Editors will have the ability to change this image dynamically.



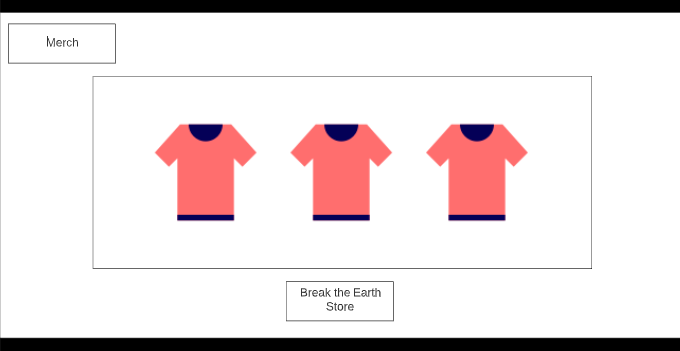
* + Now Available - This section will inform users of the current album available for purchase. The section will include a large image of the cover art for the featured album and include the album name and track list. Streaming service links will be available so users can listen to the album on their favorite streaming service. The goal here is not to limit the user on where to obtain the music, so a player will not be embedded. The Administrator and Editors will not have the ability to make changes to this section dynamically. Information in this section will be static.
  + Coming Soon - This section will inform users of upcoming music. The section will feature a round progress bar that will communicate the progress of the album. A track list will be visible with a horizontal progress bar for individual tracks. These will be clickable and drop-down a table where the user will be able to see which instruments have been recorded. Each instrument’s value to the progress bar will be 25%. All tracks in the upcoming album will add to the progress bar of the album as well as mix and master buttons up to 100%. The Administrator and Editors will have the ability to make changes to this section dynamically.



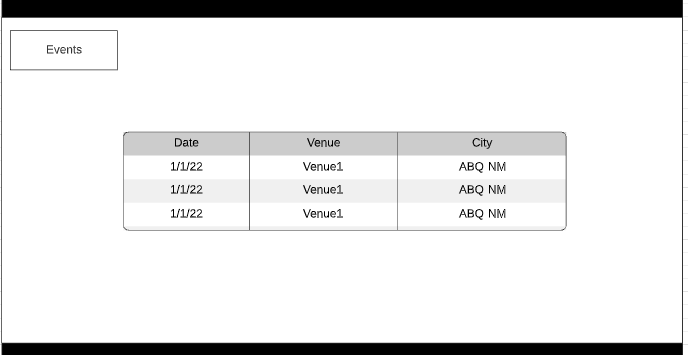
* + Videos - This section will showcase the most recent available video by the group. The video should be embedded from YouTube. The Administrator and Editors will not have the ability to make changes to this section dynamically.



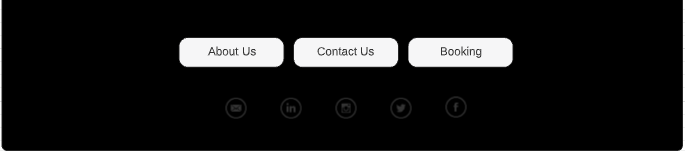
* + Merchandise - This section will showcase currently available merchandise for purchase. The client has an online store set up through TeeSpring. The web application will provide images of the store merchandise. These images will be clickable and will direct users to the purchase page for the piece of merchandise clicked. The Administrator and Editors will not have the ability to make changes to this section dynamically.



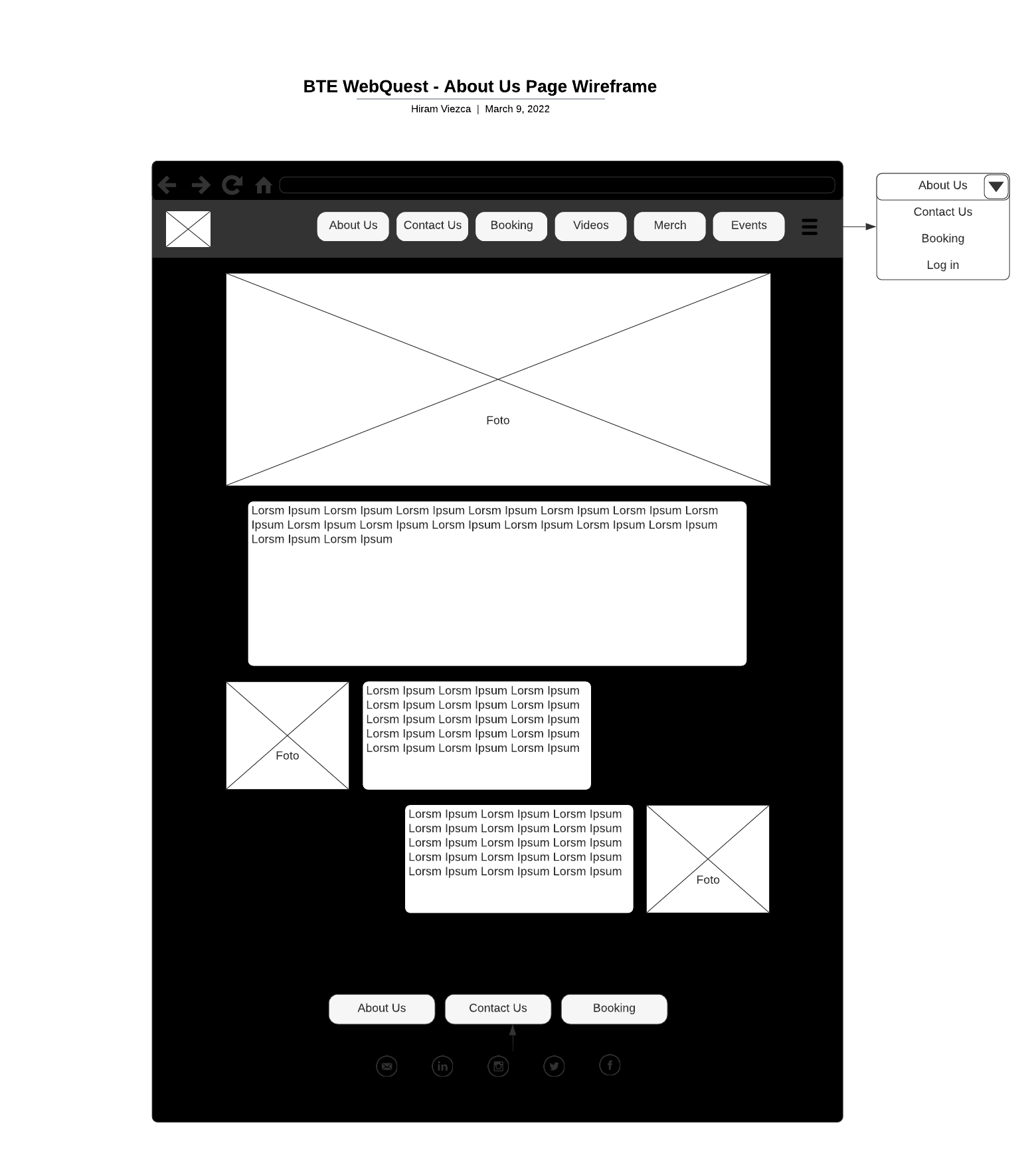
* + Events - This section will have a list of upcoming events so visitors can be informed of where they can go to see the band live. The section will feature a table that will list the venue, date, city, and state of the upcoming events. The Administrator and Editors will have the ability to make changes to this section dynamically.



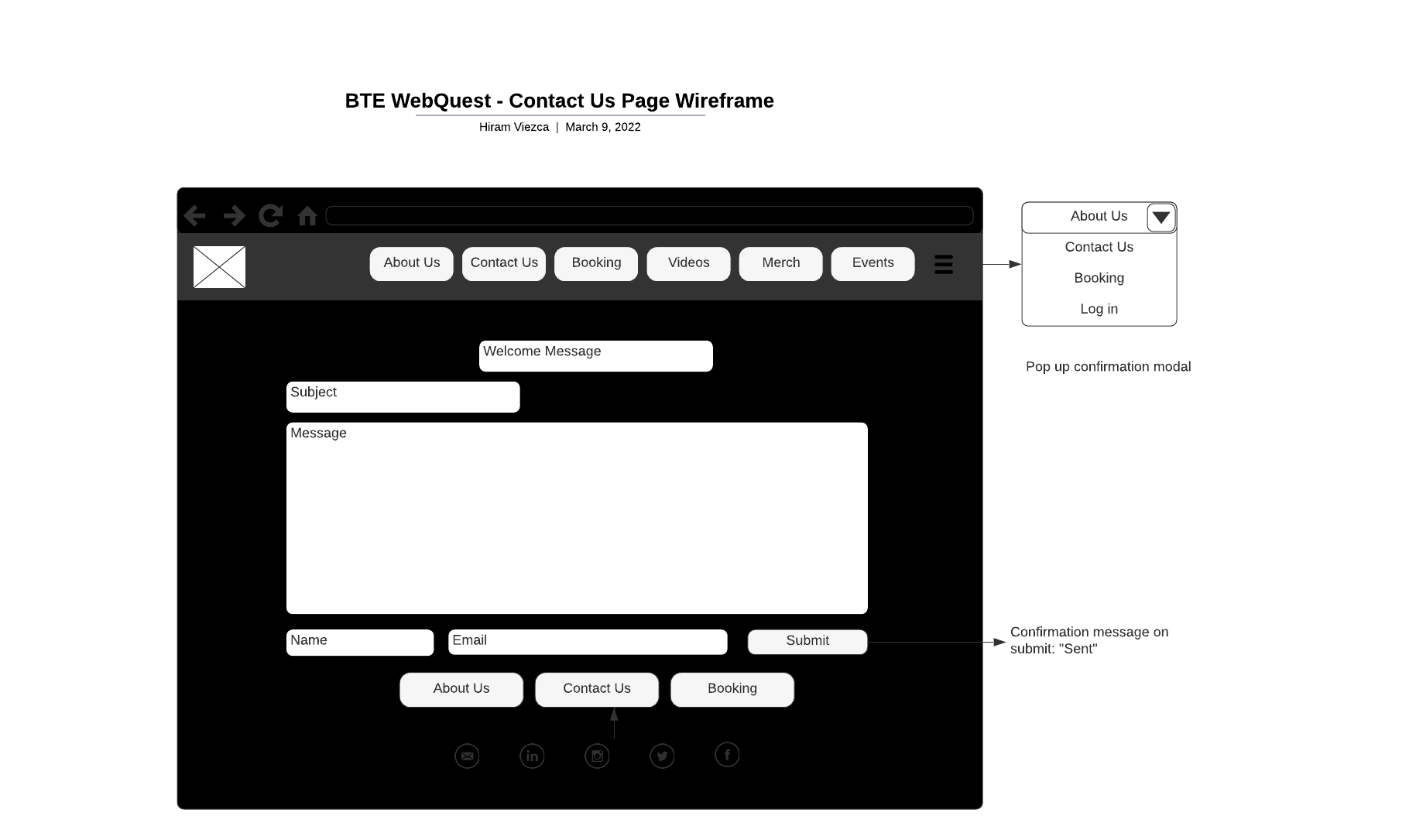
* + Footer - This section will include navigation buttons for the About Us, Contact Us, and Booking pages, as well as links to the band’s various social media sites.

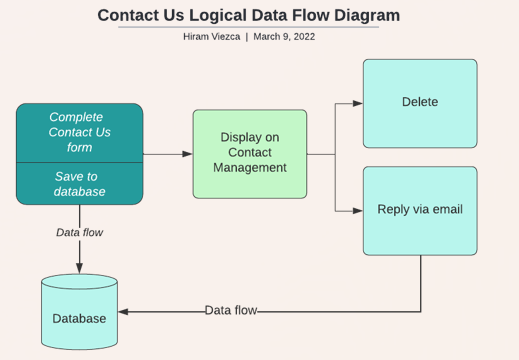


* “About Us” page - This page will have a large band image along with a biography for the group. Each band member will have smaller images and a short biography for themselves. The Administrator and Editors will not have the ability to make changes to this section dynamically.



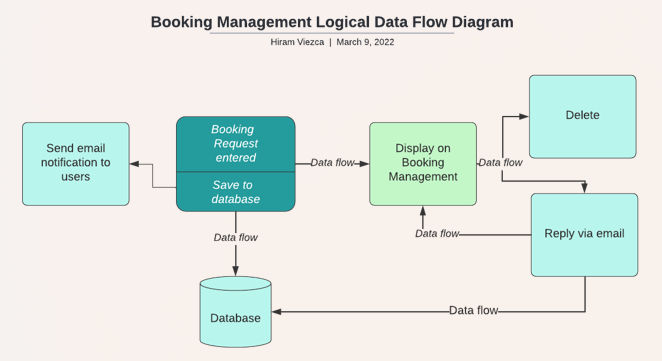
* “Contact Us” section - This section will contain a form for users to send a message to the group. When a user submits the contact request, the application will save the message in the database. The application will then send an email to the client’s designated email address via Spring Email and display it in the Administration Contact Us Management page.

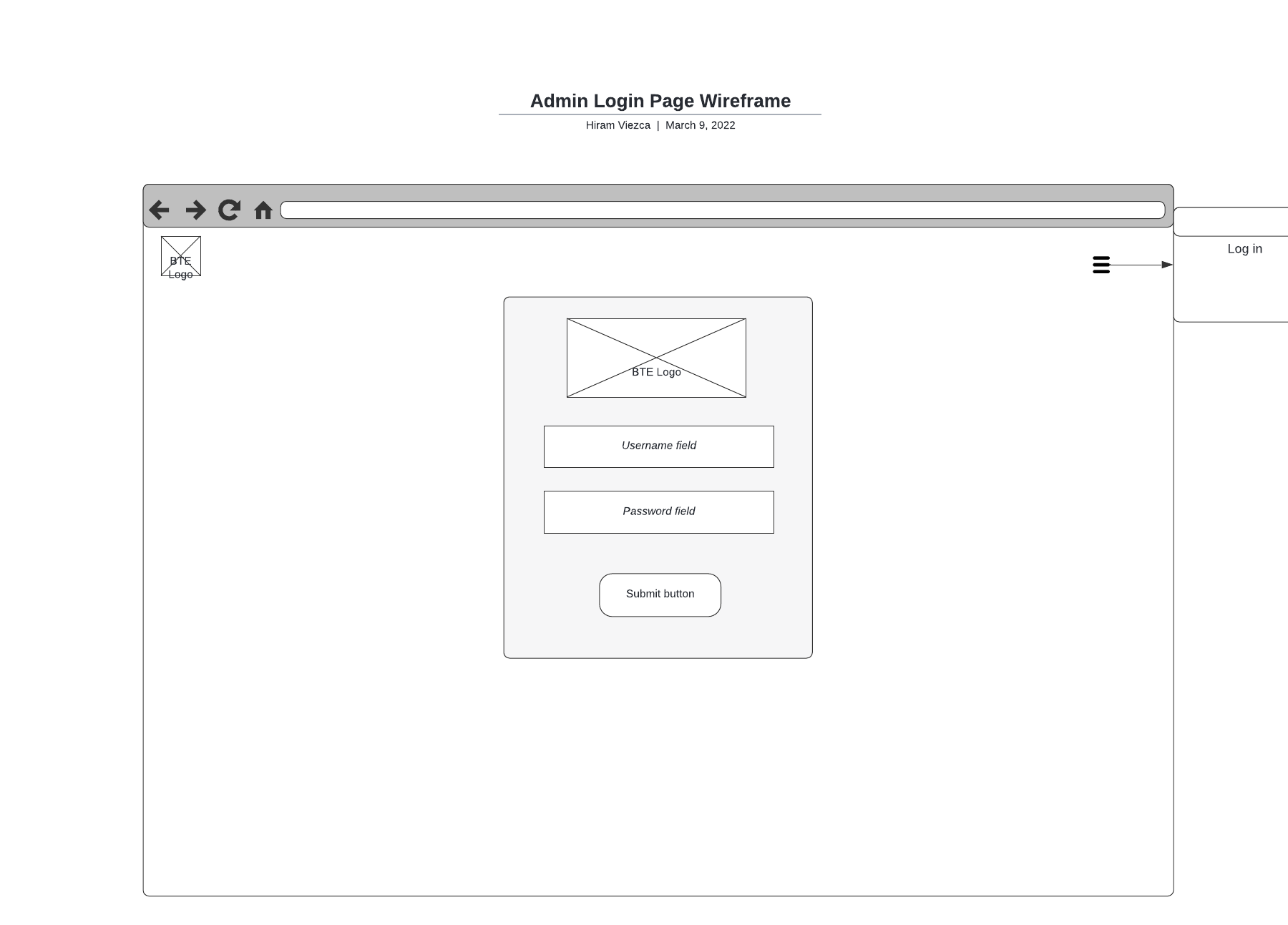


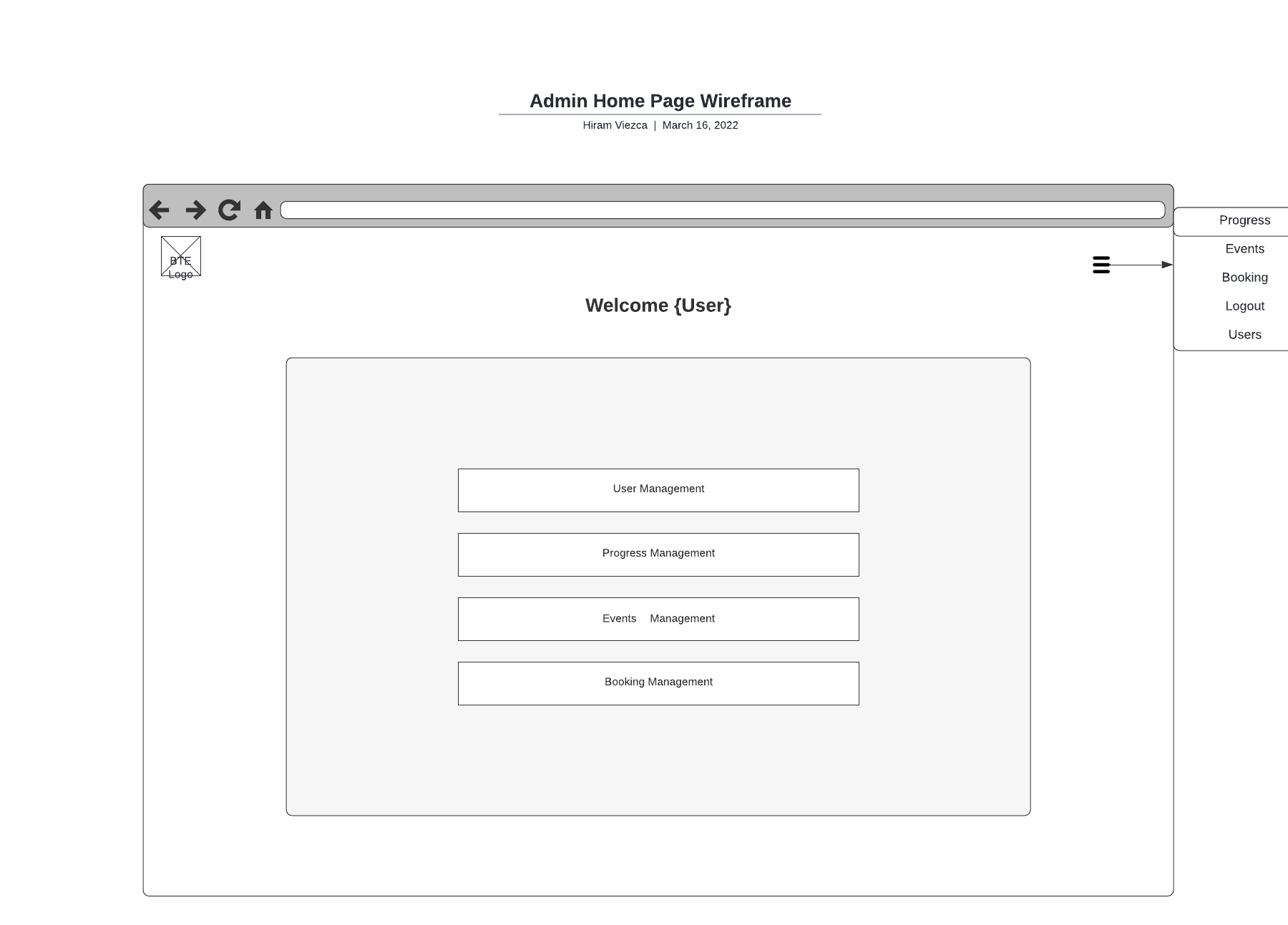


* “Book Us'' section - This section should contain a form for users such as venue representatives or booking agencies to send the group a message regarding a booking opportunity. When a user submits the booking request, the application will then save the information in the database. The application will then send an email to the client’s designated email address via Spring Email and display it in the Administration Booking Management page.

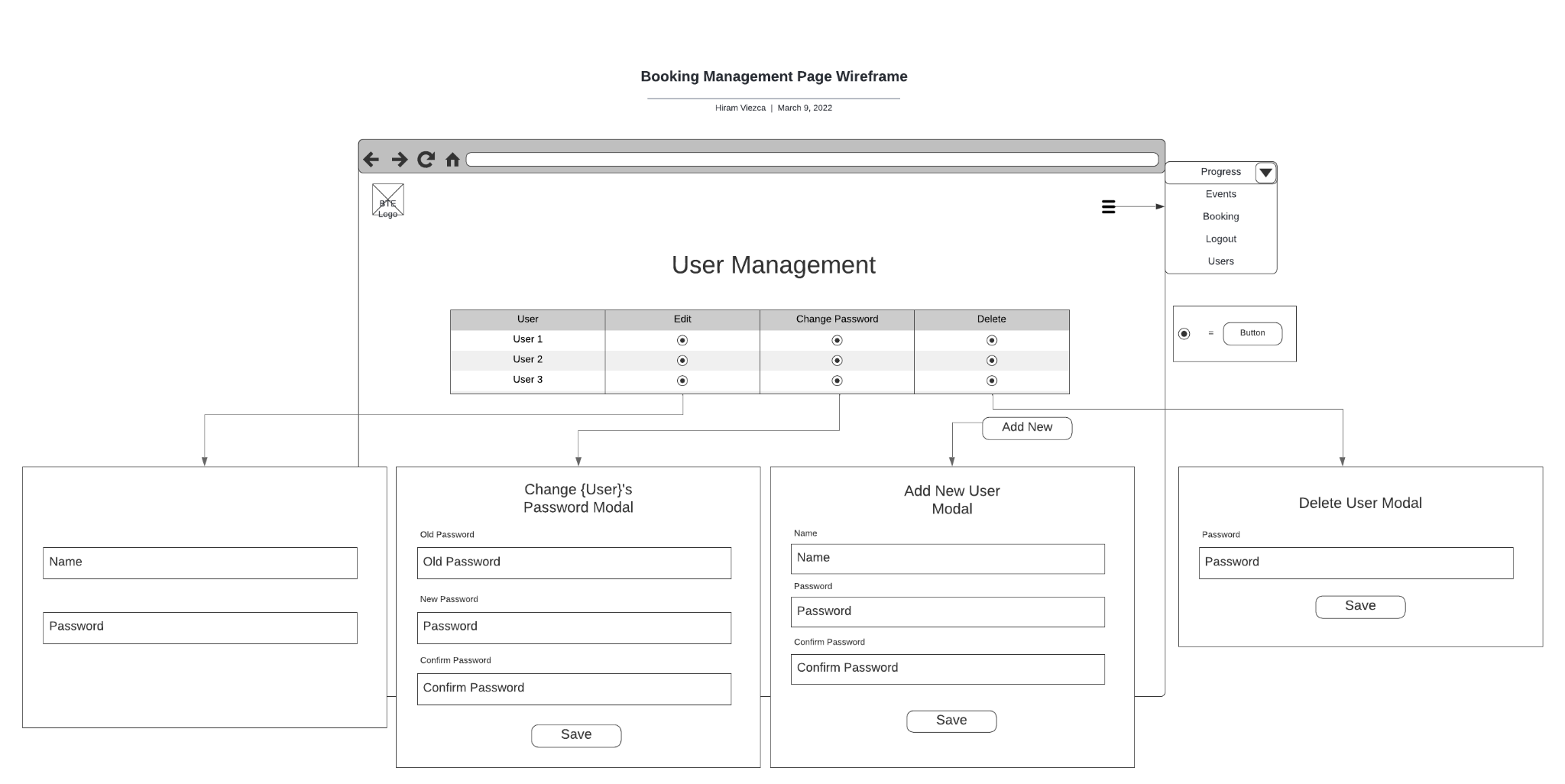




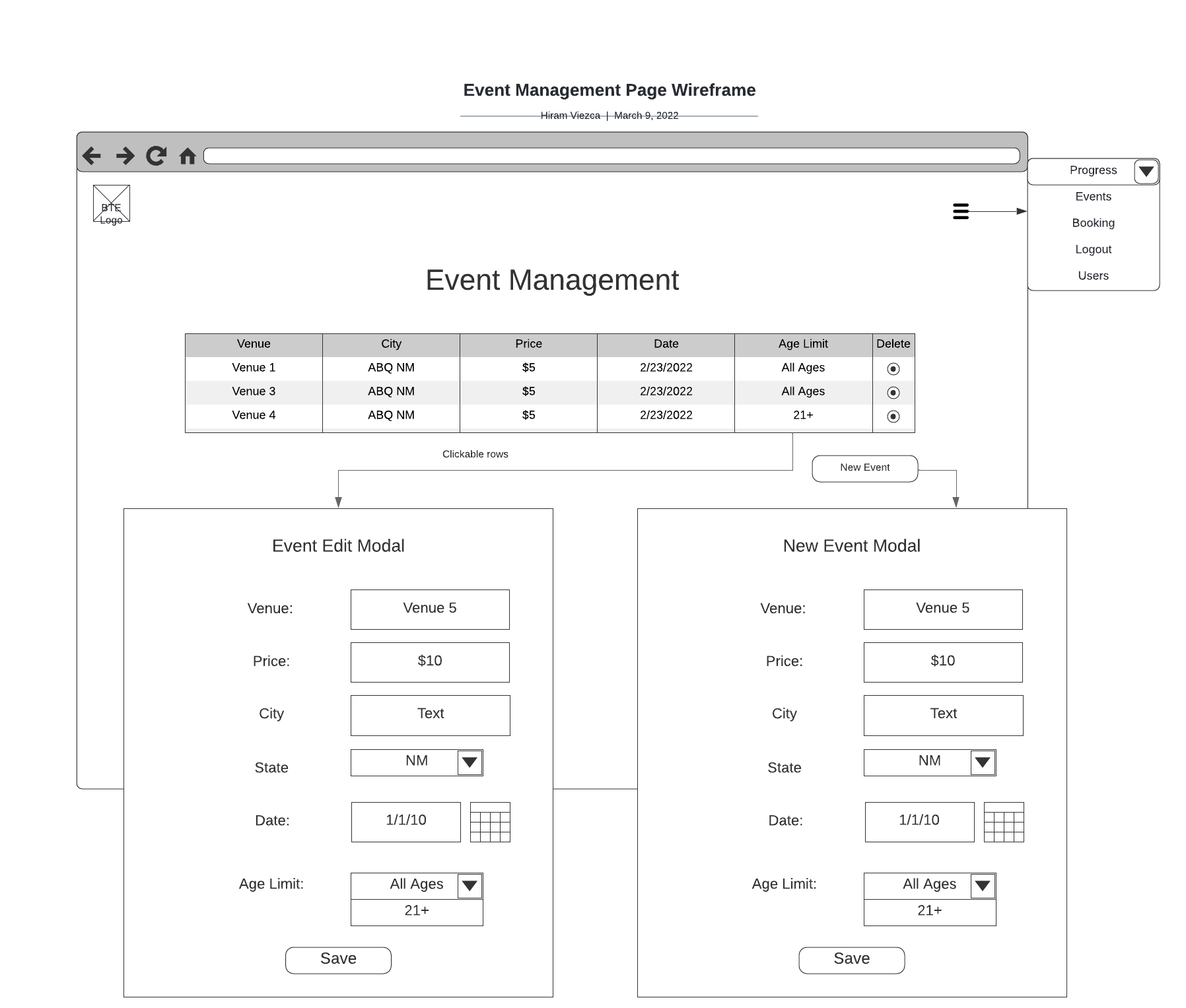
* “Administration” - The Administration section allows for the Administrator and Editors to be able to manage and update all the content on the website. The administration section provides a way to view, manage, and respond to messages sent through the “Contact Us” and “Booking” pages. The Administration section will be protected with role-based security via Spring Security. The Administration section should have the following sections at minimum.
  + “Admin Login” - This section will allow the Administrator and Editor users to log in to the administration section of the website. 
  + Admin Home - This page will serve as the landing page for Administrator and Editors after logging in. From this page the users will be able to navigate to different administrative tasks.



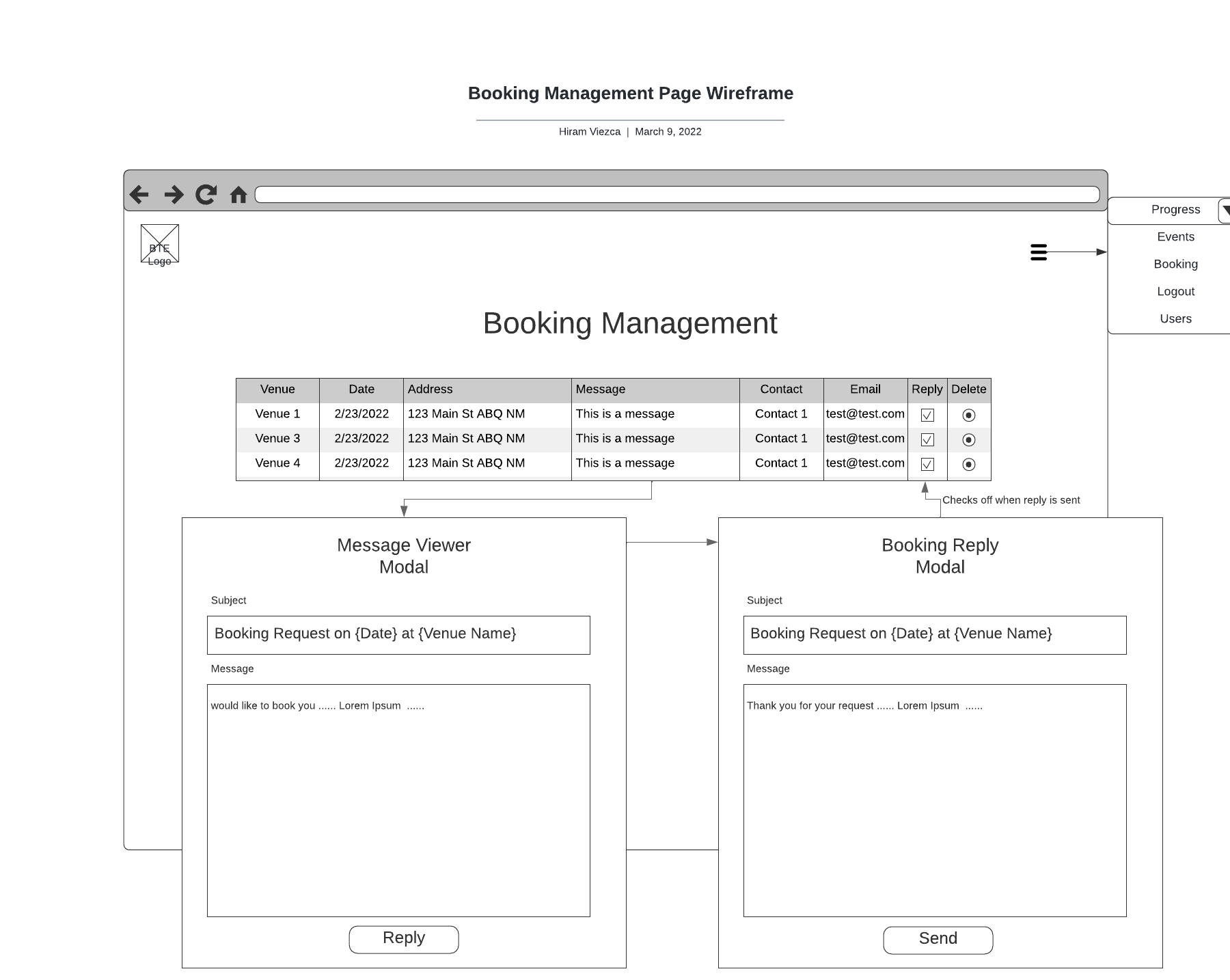
* + User Management - This section will allow the Administrator and Editors to manage user related tasks. This page will display differently depending on the user’s role. The Administrator will have full access to all functions including adding a new user and deleting users. Editors will not be able to add or delete users. Administrators and Editors will not be allowed to change usernames or passwords except their own. All functions will require the respective user’s password to complete.



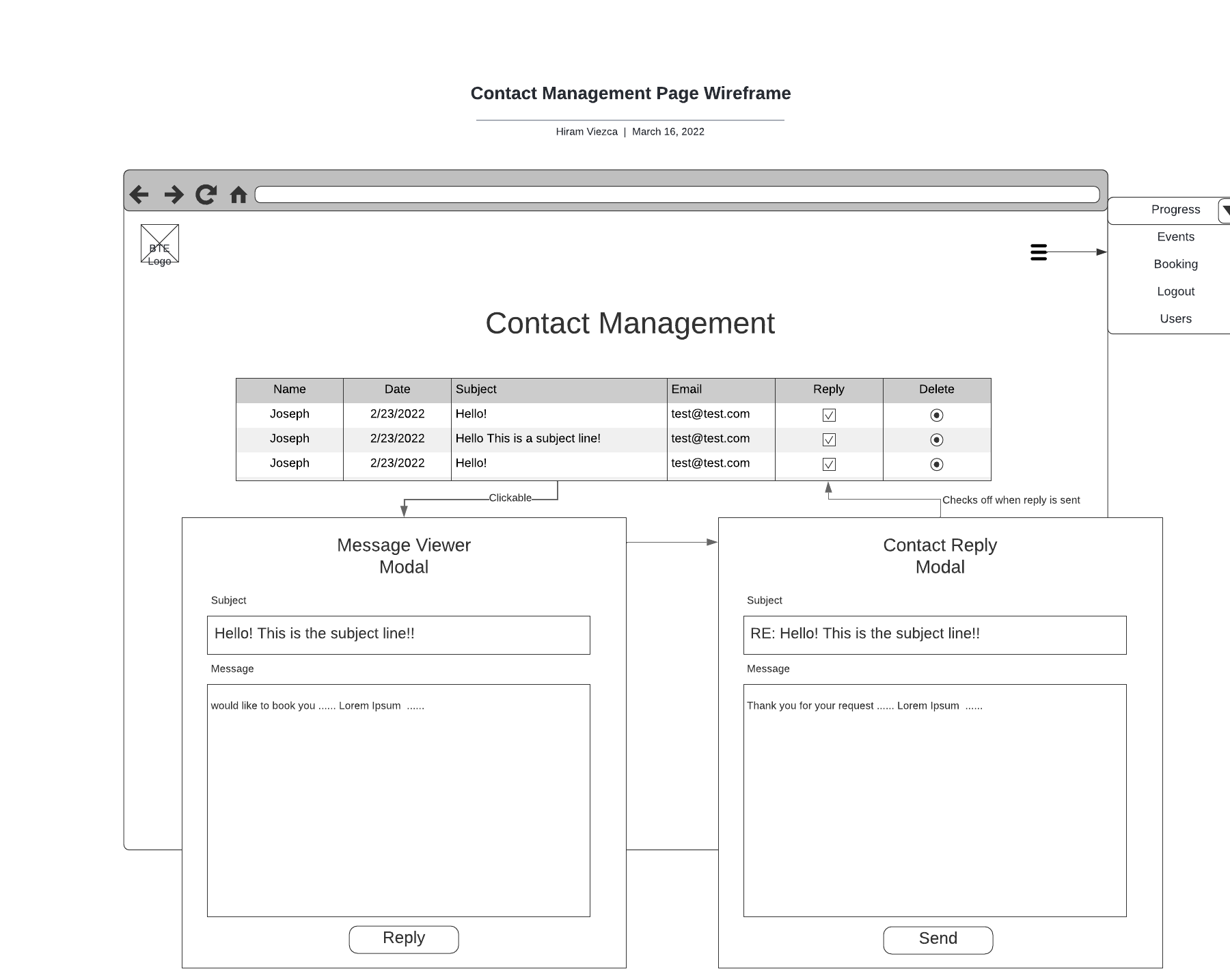
* + Event Management - This page will allow users to manage events. Administrative users will be allowed to add new events, edit existing events, and delete events. Events on the table on this page will display on the application Main page.



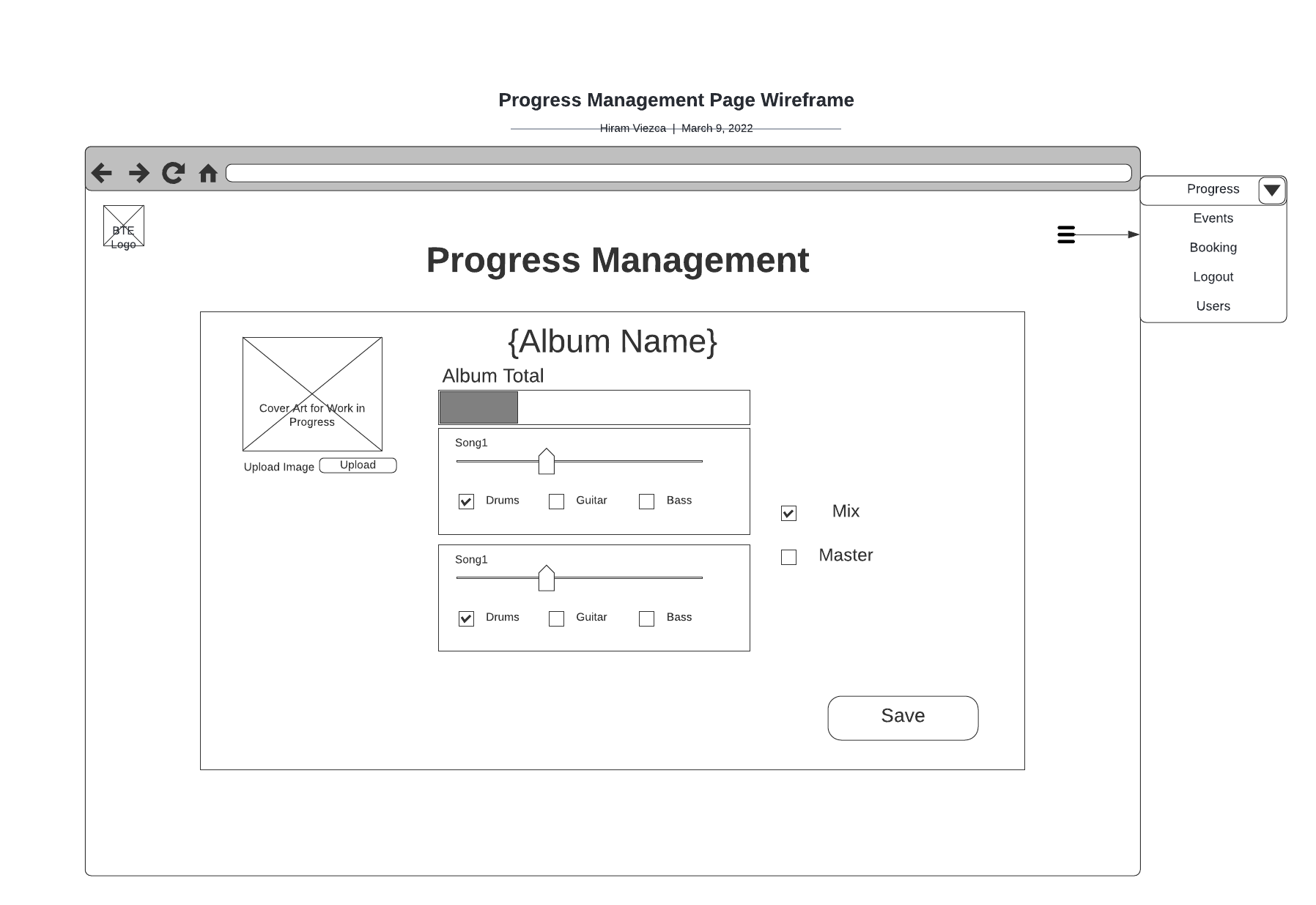
* + Booking Management - This page will allow administrative users to manage booking requests. Users will have the ability to reply to requests from this page and an email will be sent by the application via Spring Email.



* + Contact Management - This page will allow administrative users to manage contact requests. Users will have the ability to reply to requests from this page and an email will be sent by the application via Spring Email.



* + Progress Management - This section will allow administrative users to manage the new material progress section of the application Main page. This section will allow users to upload an image and manipulate progress bars through different checkboxes.



**Deliverables**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Deliverable Acceptance Log | | | | | |
| ID | Deliverable Description | Comments | Evaluator (internal or external as applicable) | Status | Date of Decision |
| 1 | Architecture Design Document | This document | Client | Completed | 3/16/22 |
| 2 | Data Dictionary | Defines data to be persisted in database | Developers | Completed | 4/11/22 |

**Detailed High-Level Solution Design**

**Logical Solution Design:**

Login/User Administration with Roles

User and role data will be stored in either a MySQL or PostgreSQL database. Interfacing to the database with Spring Boot will be done with Spring Data JDBC. Spring Data allows us to use an Object Relational Mapping (ORM) approach that can be leveraged with either database we choose to use. The flow of data will start with user data passed from the view via a UserModel to the controller and translated to a UserEntity in the business service. The UserEntity will be passed to the database through the CRUD repository with the process going in reverse to deliver data back to the view.

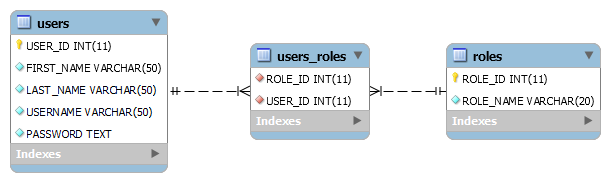
Login and user administration will both use the same model and controller to accomplish their respective operations. The UserModel will contain basic data to include the first name, last name, username, and password of the individual. This information is kept basic as only members of the band will have access to the extended functionality provided through authenticated access.

Roles will receive their own RoleEntity class to store their data when retrieved from the database. There will be two roles, ‘Admin’ and ‘Editor. Roles will be linked to the user’s id in the relational database and pulled upon login. Spring Security will use these roles to control access to URLs and sections of HTML code.

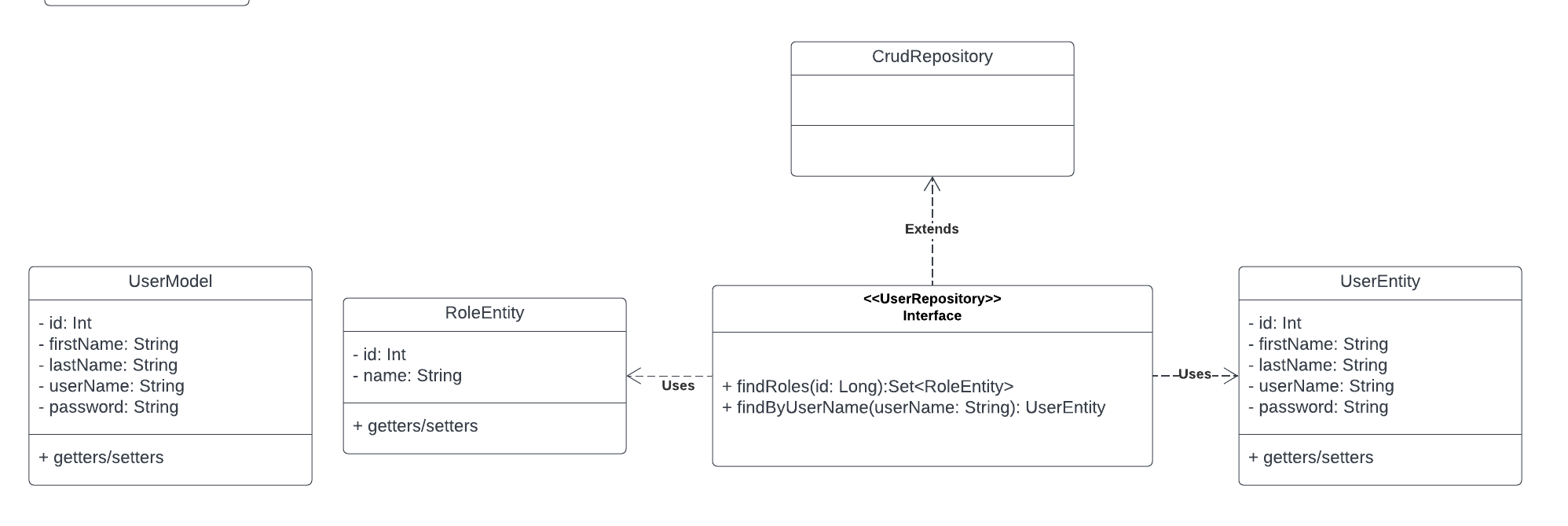
The ‘User Management’ page will provide users with the ‘Editor’ role the ability to change their own password and user details. Users with the ‘Admin’ role will have the ability to both create and delete users.

The Login Page will allow the user to enter their username and password to be verified against users in the database. The logic for validation will be handled by Spring Boot Security allowing for the usage of roles. When the user is validated, their applicable roles are loaded for reference when accessing data and URLs.

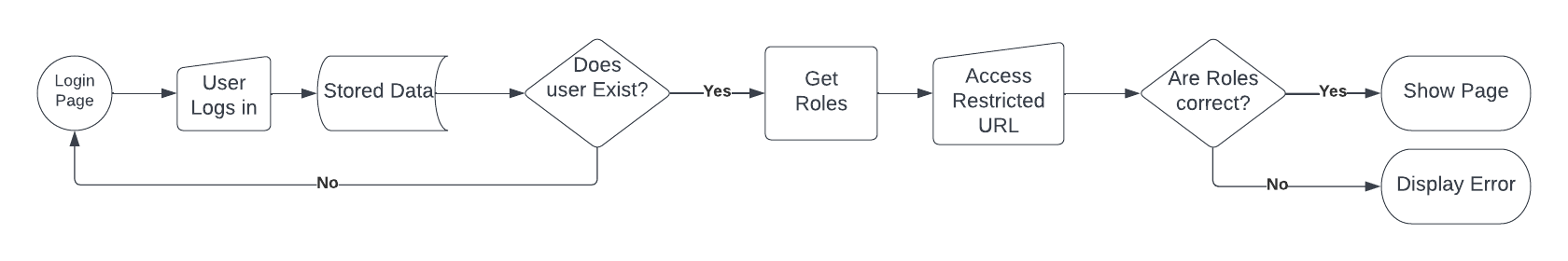
ER Diagram

****

Class Diagram

****

Process Flow



**Main Site Page – Overview**

The Main Page will provide a mix of static and dynamic information. The current design separates the main page into multiple sub-sections to include:

* Now Available (Static) - Information on and links to music from the bands current catalog,
* Coming Soon (Dynamic) - Information on and status of progress concerning upcoming material from the band.
* Videos (Static) - Embedded online videos of the band.
* Merchandise (Static) - Carousel of band merchandise for sale and links to online stores.
* Events (Dynamic) – Listing of events where fans can see the band.

The subsections of the main page will be coded as separate Thymeleaf fragments allowing them to update separately from the main page. The use of fragments also allows the removal of sections or addition of new sections without redesigning the main page.

The static pages will have information that is hard coded into the HTML and will require updates to the pages themselves. The dynamic pages will pull data from the database to ensure the most current data is available. These sub-sections will be discussed in more detail below.

**Coming Soon Display and Management**

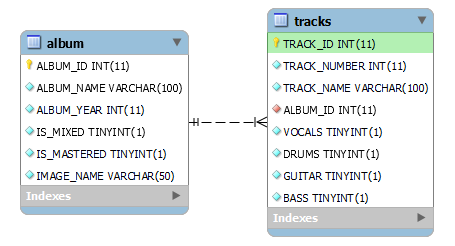
The ‘Coming Soon’ section will provide insight into the completion of the band’s current projects. This section will be broken down into progress on individual tracks and the album. Each track will display the name and progress of each of the four major parts involved in its creation, drums, guitar, bass, and vocal. All track completion percentages will feed a circular progress bar indicating album completion.

Albums, tracks, and their progress status will all be stored within the database. Updates to this data will be accomplished from the ‘Progress Management’ page. The ‘Progress Management’ page will provide users with the ‘Editor’ role the ability to open the current album and display all track’s progress. Each track will have an editable name and radio/check buttons to toggle completion of the different instruments for that track. An overall ‘Mix’ and ‘Master’ radio/check buttons will be available for toggle as well. A ‘Save’ button will be provided to update the data in the database.

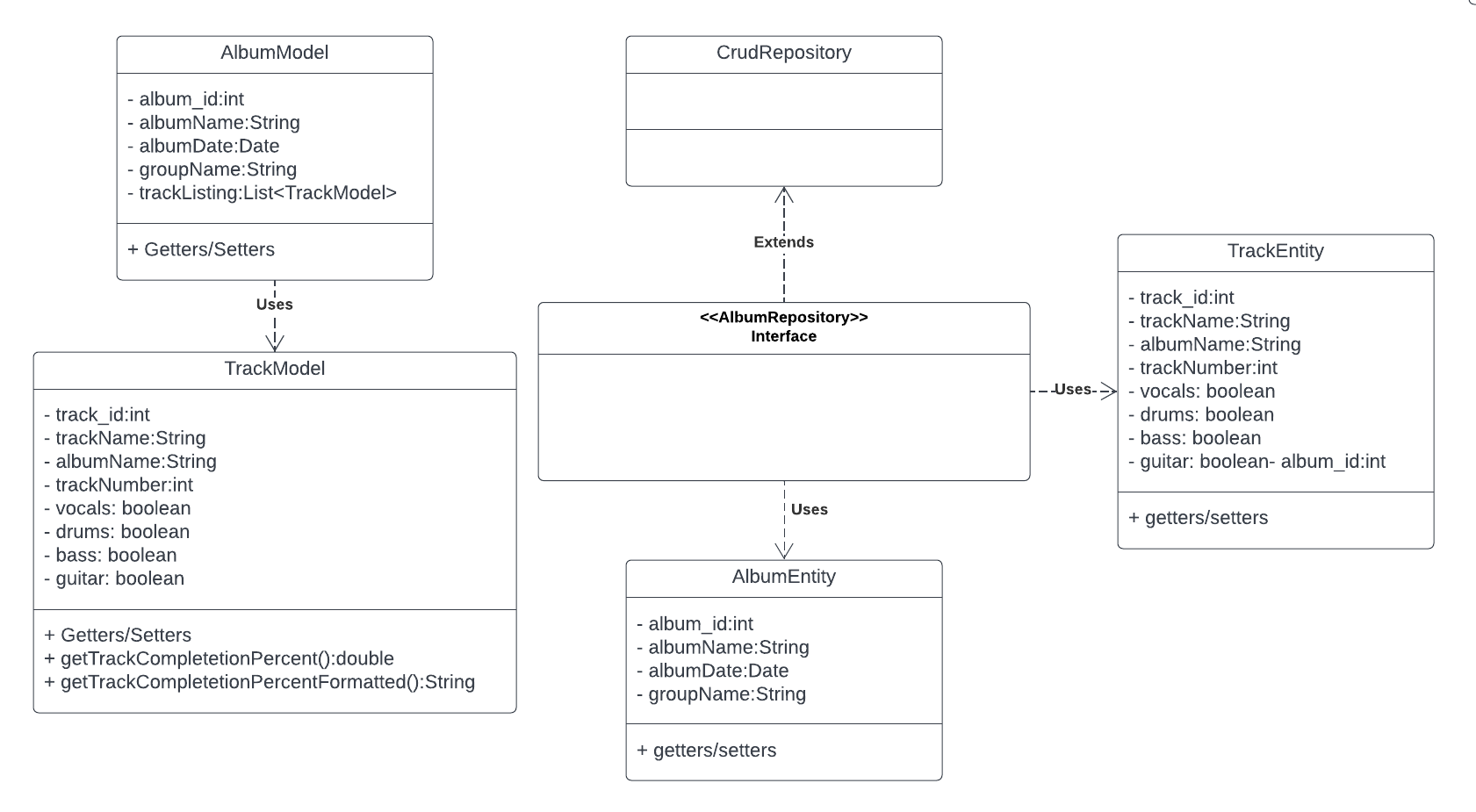
Data from the ‘Progress Management’ page will be saved in an AlbumModel that will contain a List of TrackModels representing each track. Each TrackModel will contain the status of each track's progress data. The AlbumModel will be passed to the AlbumBusinessService where they will be separated into an AlbumEntity and TrackEntity and passed into the database through a CRUD repository.

When the main page is requested, the current project album along with its tracks and their progress will be pulled from the database by the AlbumController via CRUD repositories. The retrieved data will be converted to an AlbumModel. This AlbumModel will be sent to the view to display the most current information available. The progress bars will be coded with a mix of Bootstrap and JavaScript.

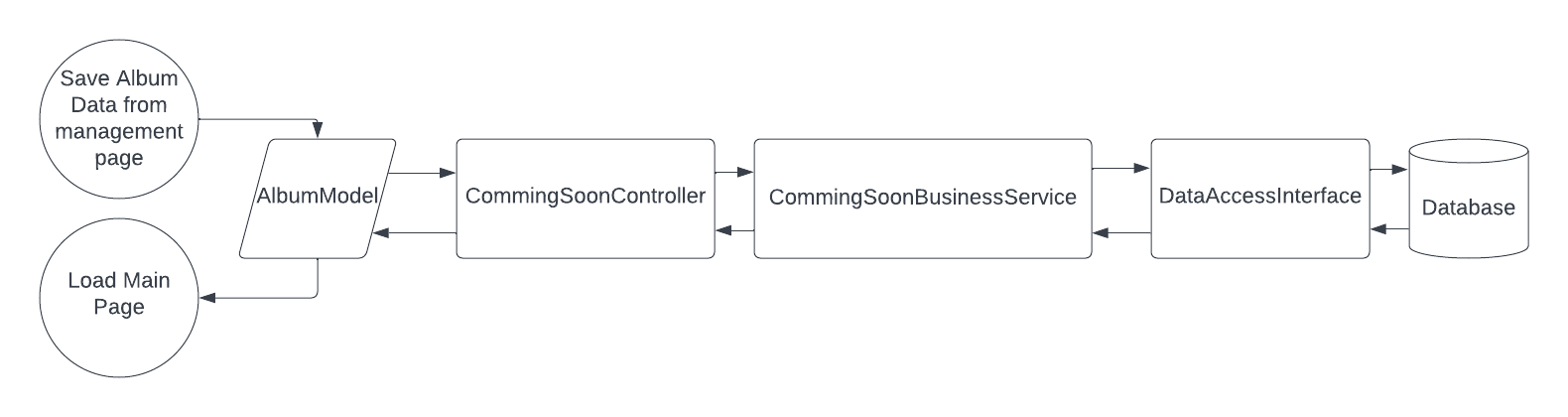
ER Diagram



Class Diagram



Process Flow

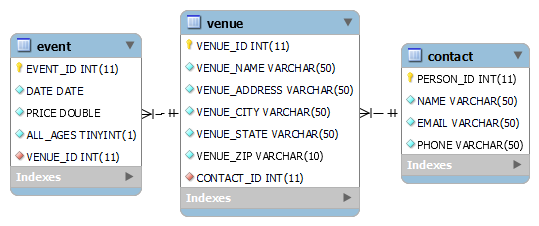


**Main Site Page - Events**

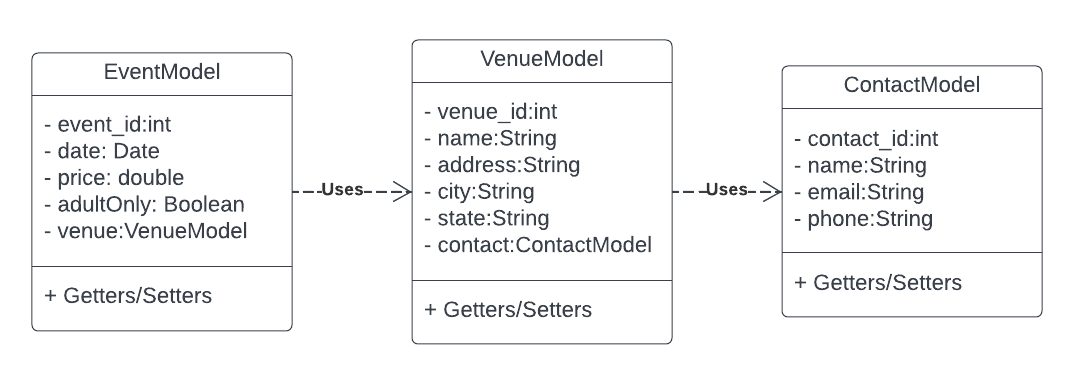
The ‘Events’ sub-section of the ‘Main Page’ will contain a list of upcoming events. These events will be entered by a registered user on the ‘Event Management’ page or through a Booking Request.

Each Event will be a separate item within the database and will be inserted and removed with an EventEntity that will contain all applicable information for the event. The EventEntity will be linked to a VenueEntity which in turn is linked to a ContactEntity. The EventEntity, VenueEntity, and ContactEntity will be collected into an EventModel in the EventBusinessService. The EventBusinessService will provide the methods for dealing with an event and provide a method that will produce a List of EventModels matching date criteria. This List of EventModels will be used to populate the ‘Events’ section of the main page.

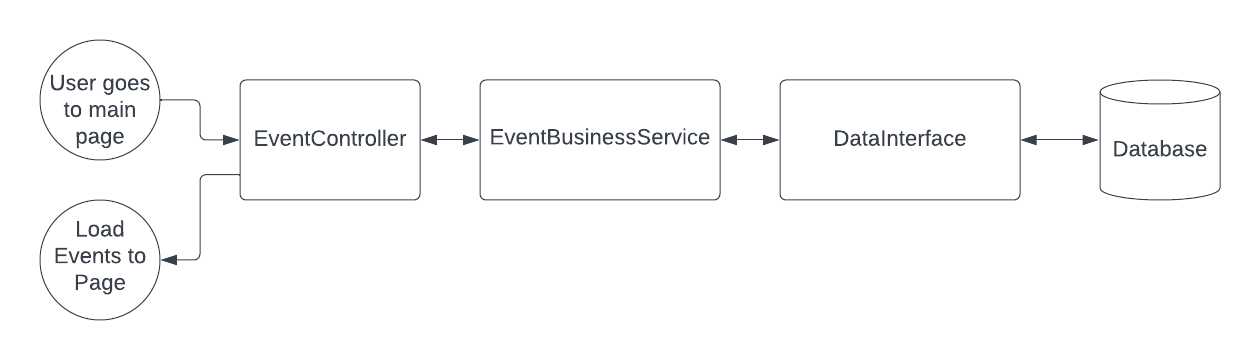
ER Diagram



Class Diagram



Process Flow



**About Us Page**

The ‘About Us’ page is a static page with information about the band and photos. This information will not change very often as it is a general overview of the band. This page will not require any coding and consist of HTML

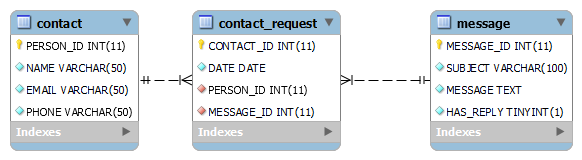
**Contact Us Page and Management**

The ‘Contact Us’ page will provide a form that collects pertinent information from the site user and submit it to the website administrator and editors for review. The form will ask for the subject, message, user’s name, and email address. Upon submission of the form, the web application will send an email to the client’s designated email address informing of a new contact request received.

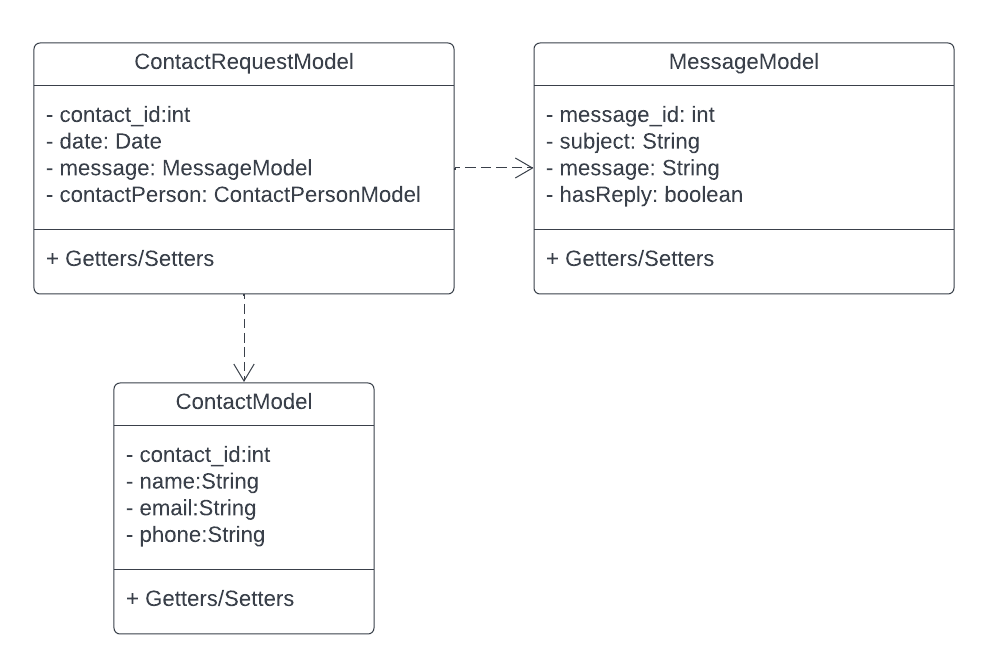
When the message is submitted the data will be collected in a ContactRequestModel. The ContactRequestModel will contain all the information for the message and the user and sent to the ContactController. The ContactController will convert the model to separate entities for the Request, Message, and Contact and use the applicable DataAccessInterface to insert this data into the database.

The ‘Contact Management’ page will provide users with the roles of ‘Editor’ or ‘Admin’ the ability to respond to contact requests. The ContactController will populate a list of ContactRequestModels from the database. This list will be used to populate a table with all available messages. The table will provide the name of the sender, date, subject, sender’s email, a check box indicating if a reply has been sent, and a delete button. The user can click on the subject column to open a modal with the entire message. This modal will provide a button that will open another modal allowing them to construct an email in response. When the email is completed, the has\_reply information will be updated.

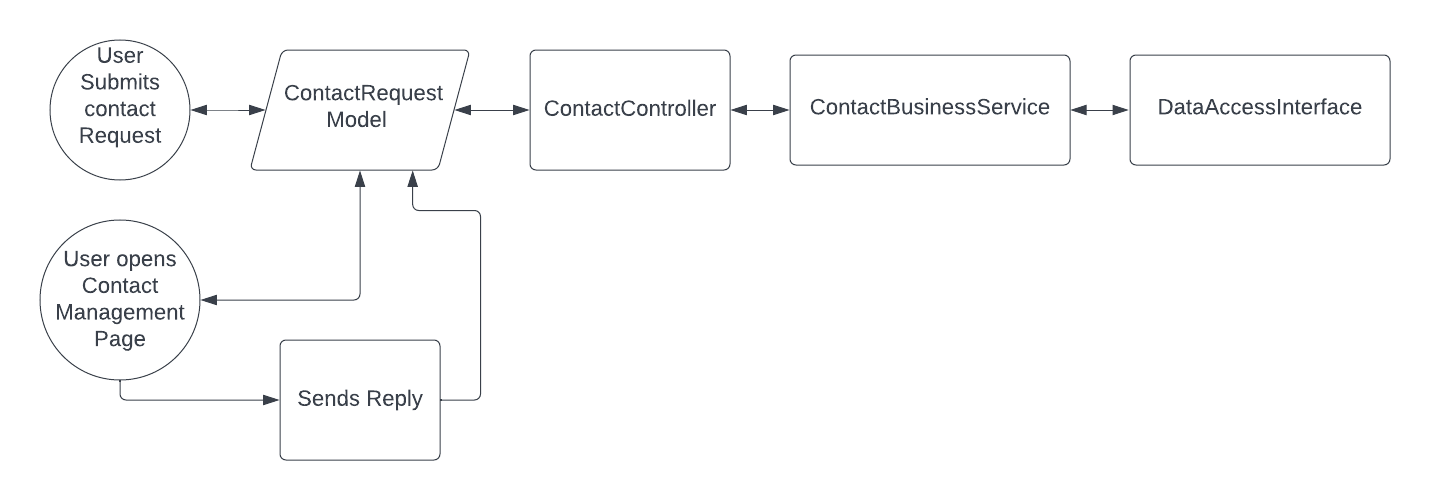
ER Diagram



Class Diagram



Process Flow



**Book Us Page and Management**

The booking process starts with the ‘Book Us’ page. This page will provide the opportunity for businesses and booking agents to contact the band for a show. Booking requests will be accomplished by filling out a form with the following information:

* Venue
* Name
* Date
* City
* State
* Zip
* Message
* Contact
* Name
* Contact
* Email
* Contact Phone

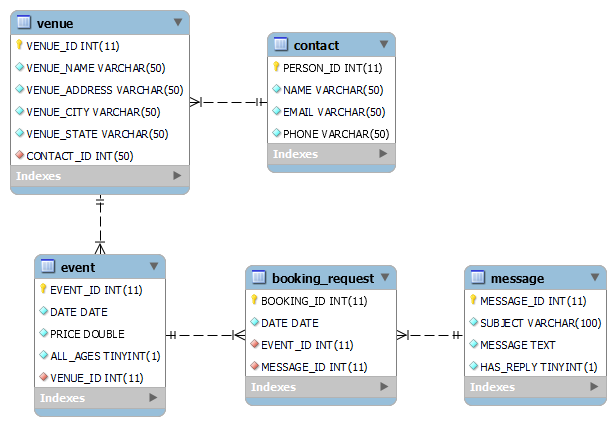
The booking request will be sent as a BookingRequestModel that will contain all form data to be stored in the database. The BookingRequestBusinessService will create an individual EventEntity, VenueEntity, ContactEntity, MessageEntity, and BookingRequestEntity from the supplied information. These entities will be stored in the database with their applicable CRUD repositories. Upon completion of the form and submittal, a receipt email will be sent to the requesters provided email.

All submitted booking requests will be managed through the ‘Booking Management’ page. This page will be available to all registered users with the ‘Admin’ or ‘Editor’ role. The ‘Booking Management’ page will display a table of booking requests with the following information:

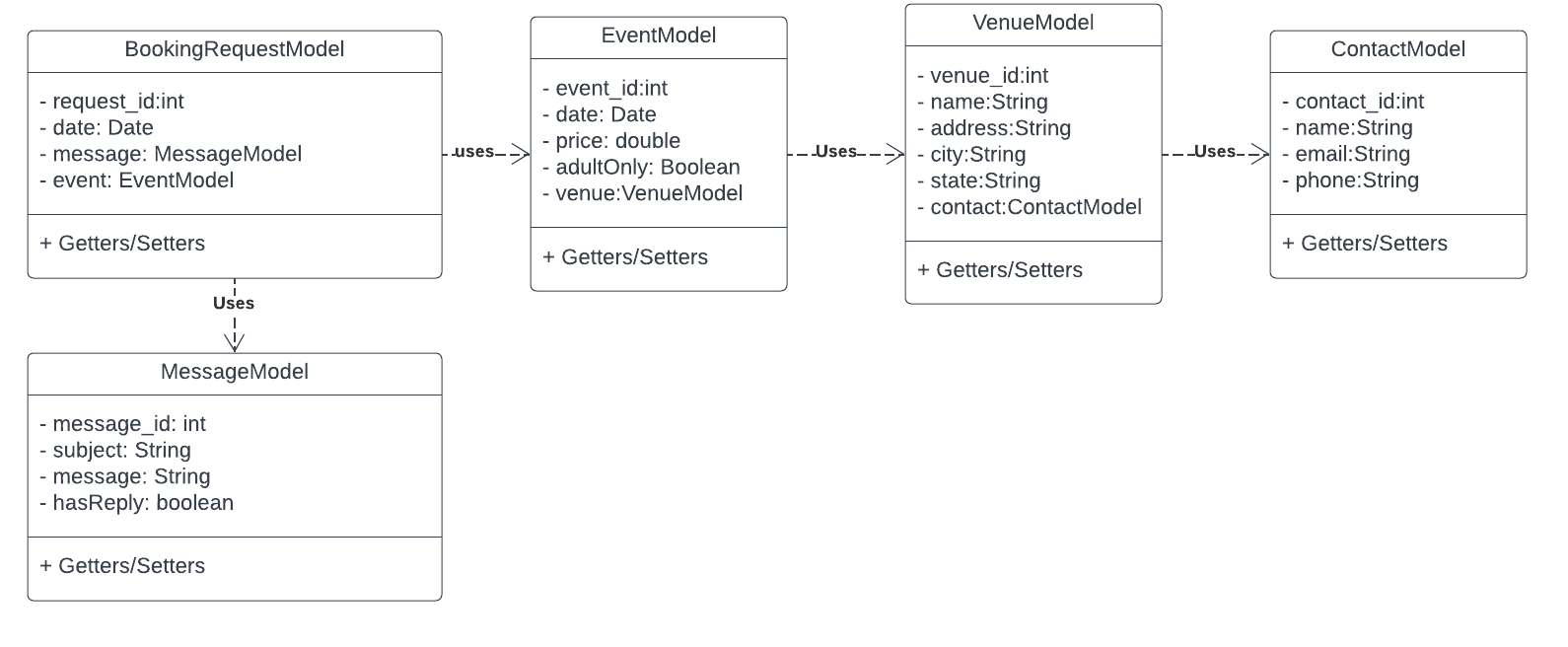
* Venue Name
* Date of Event
* Address of Venue
* Message from request
* Event contact’s name
* Event contact’s email
* Status of reply
* Delete request button

Information for the table will be pulled as a List of BookingRequestModels from the database. The user will be able to select the message and view it in its entirety within a pop-up modal. The pop-up modal will have a reply option that will bring up a new modal for replying to the message. This reply will be sent as an EmailModel to the EmailBusinessService and will be sent to the provided contact. Once an email has been sent, the ‘has\_reply’ will be updated for the request in the database. The management page will also provide the ability to delete a request.

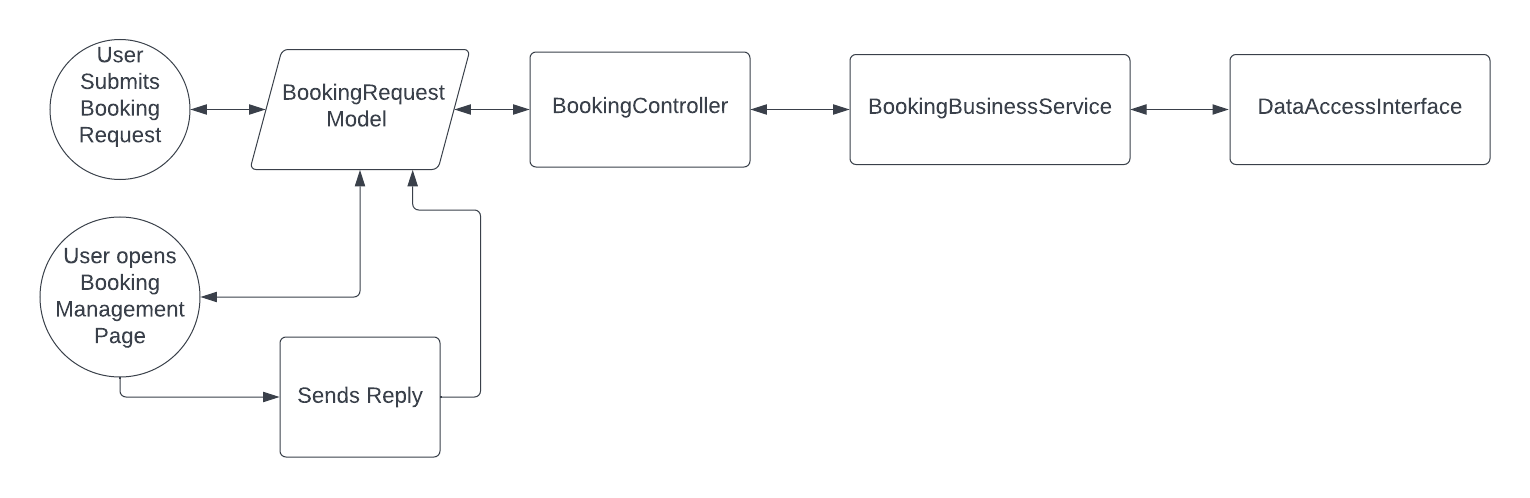
ER Diagram



Class Diagram



Process Flow

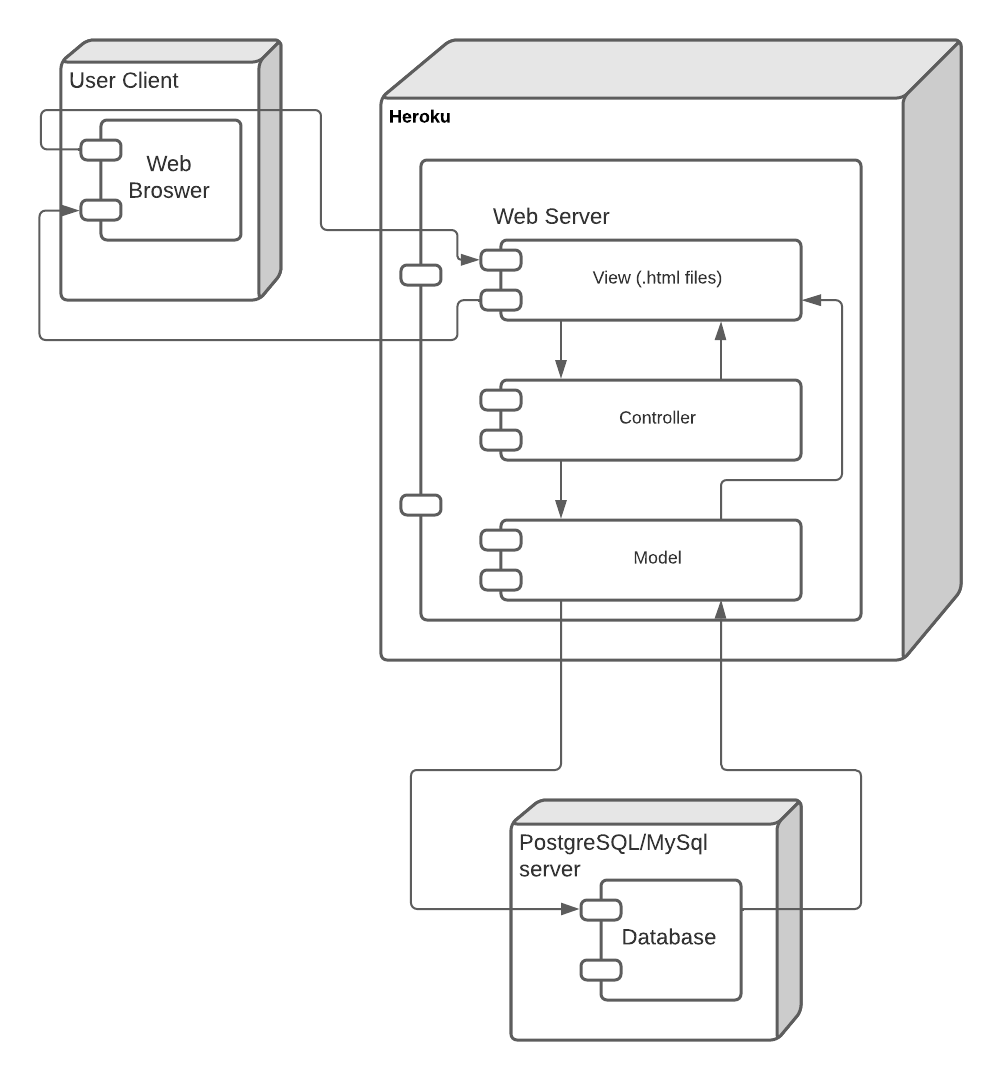


**Email**

Emails will be sent through the application in several instances. These instances include an external user submitting a contact request or booking request and an internal user responding to user contact requests and booking requests. Emails will be handled with Spring Boot Starter Email and employ an email address specifically made for these interactions.

**Physical Solution Design**

This application is designed to be a web application hosted on a cloud server. During development the team will host the project on Heroku’s free tier of web hosting. Heroku’s free tier also provides a free PostgreSQL server that will be used for data storage. This approach will provide the ability to test the application in a real environment. The application is being developed in Java with the Spring Boot framework. This language and framework was selected for its compatibility with many cloud service providers. The client will research available cloud providers and select one fitting their price point.

****

**Non-Functional Requirements**

NFR-1

Role management will be handled automatically by the application. The client only wants one ‘Admin’ user for the application. To address this, the DDL script will be set up with an initial user who has ‘Admin’ access. Each additional user created within the application will be assigned the ‘Editor’ role.

NFR-2

Physical security of the application and the cloud server it will be stored on falls upon the cloud service provider chosen by the client. The client will research security and redundancy capabilities of the cloud service providers before selecting one.

NFR-3

Roles are assigned automatically by the application and cannot be updated or assigned by any user preventing them from being able to change theirs.

NFR-4

Form validation will consist of both client side and server-side verification in a combination that ensures security and performance. Client-side verification will consist of ensuring that fields are not null and verifying that passwords are entered correctly twice. Further server-side verification will verify that passwords are correct and user names are unique.

NFR-5

The use of Roles for security in conjunction with the Thymeleaf template engine provides the ability to limit code access to specific URLs and sections of HTML. These security measures will be used to restrict access to administrative pages to only authenticated users. Within the administrative pages specific actions will be limited to those with the ‘Editor’ role as applicable.

NFR-6

The use of Roles for security in conjunction with the Thymeleaf template engine provides the ability to limit code access to specific URLs and sections of HTML. These security measures will be used to restrict access to administrative pages to only authenticated users. Within the administrative pages all actions will be available to the user with ‘Admin’ role

**Proof Of Concepts**

|  |  |  |
| --- | --- | --- |
| Proof of Concepts | |  |
| **Description** | **Rationale** | **Results** |
| 1. Spring Boot Email | Ensuring that we can implement email into the application and it will send emails in accordance with ou | The results were favorable. We were able to implement email in accordance with the project needs. |
| 2. Ajax page loading | Incorporating the use of Ajax to load pages without a whole page refresh. This makes the page seem more professional with a seamless design. | We were able to incorporate Ajax into the project and perform partial page refreshes. |

**Hardware/Software Technology**

|  |
| --- |
| Hardware and Software Technologies |
| 1 - Java |
| 2 - JavaScript |
| 3 - jQuery |
| 4 - Spring Boot |
| 5 - MySQL |
| 6. PostgreSQL |
| 7. BootStrap |
| 8. Spring Tools |
| 9. Thymeleaf |
| 10. Heroku Cloud Service |
| 11. GitHub |
| 12. Jira Software |
| 13. Google Docs |
| 14. LucidChart |
| 15. JavaScript Progress Bar |
| 16. Font Awesome |

**Detailed Technical Design**

**General Technical Approach:**

This project will be a Spring Boot web application for a band. The client desires a clean, minimized, web application with the ability to easily update information being presented to fans. The application needs to consist of a main page that provides the bulk of available information to the user. Along with the main page there will be several additional pages providing communication for users and industry representatives to reach out to the band. All data pertaining to the site will be stored in a SQL database for persistence.

The team will use Spring Boot and Thymeleaf to create a dynamic main page with subsections that can be updated individually from special management pages. These management pages will also provide authorized users the ability to manage users, booking requests, and contact from users of the application.

Security will be implemented with Spring Boot Security and the use of roles. Roles will allow the team to lockout URLs and sections of code from unauthenticated users and those without identified roles.

The team will be using Agile project management to control the flow and execution of the project. This will be supported by Jira tools to help with creating epics, user stories, and sprint management. The team will be meeting several times each week for the Agile approach as seen in the below schedule.

Sunday - Sprint Planning

Wednesday - Weekly Standup

Saturday - Sprint Retrospective

**Key Technical Design Decisions:**

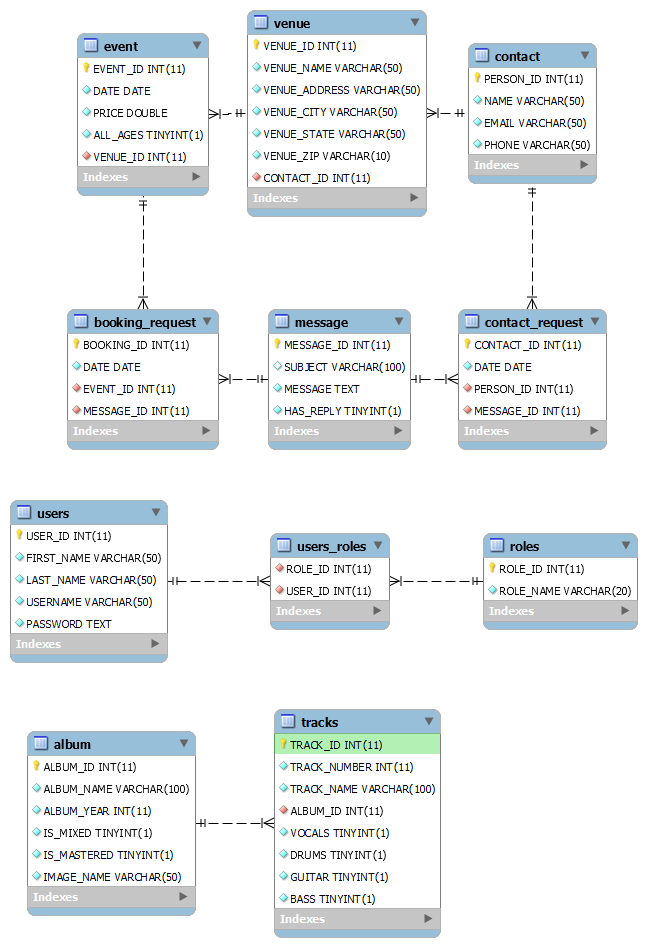
The application will be written in Java through Spring Boot as this is supported by many cloud service providers. During development, the application will be hosted on Heroku’s cloud service in their free tier. The team will use a MySQL database as well through MAMP, which the site will use for data persistence. Security will be handled as role-based security through Spring Security. This will allow only authorized users to access Administration pages.

The application user interface will be written in HTML 5 and use Bootstrap for CSS. The interface will be broken out into Thymeleaf fragments so individual sections can be updated separately and dynamically from one another. This will allow the website to have a modern look and feel.

The Administration pages will employ use of modals for updates, adding new objects, and deleting. This will be accomplished through Javascript and JQuery so page refreshes will not be necessary.

The application will include an email feature for the Contact Us and Book Us administration pages. This will allow authorized users to reply to contact and booking requests. This email feature will be handled by Spring Boot Email.

**Database ER Diagram:**

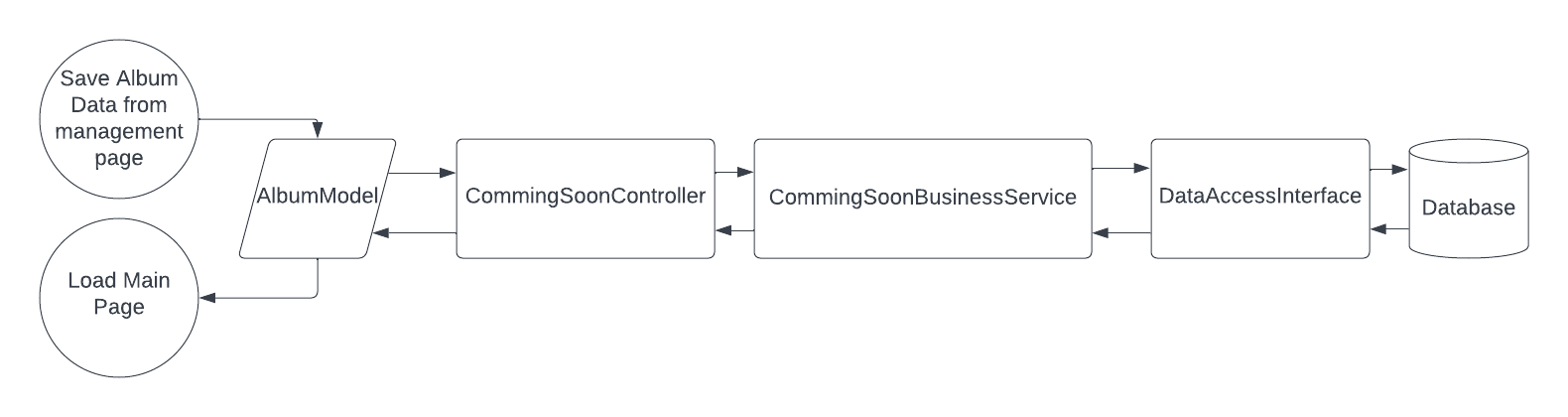


**Database DDL Scripts:**

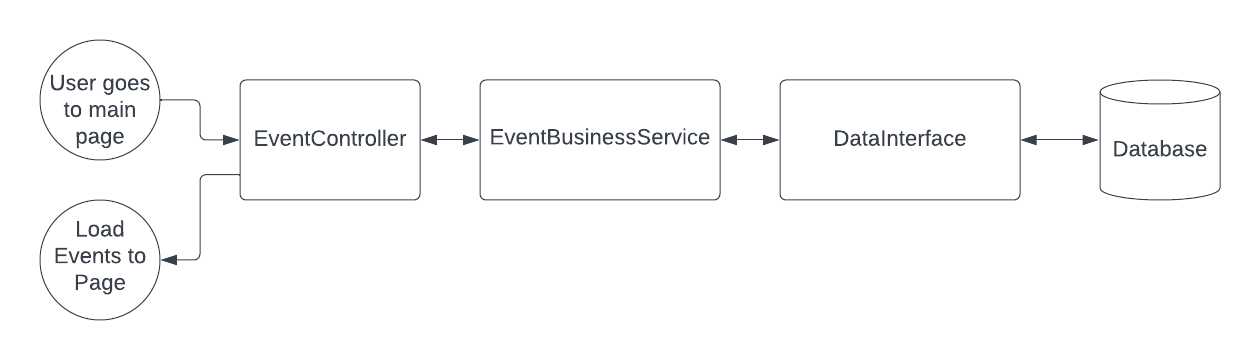
Will be available in the next milestone.

**Flow Charts/Process Flows:**

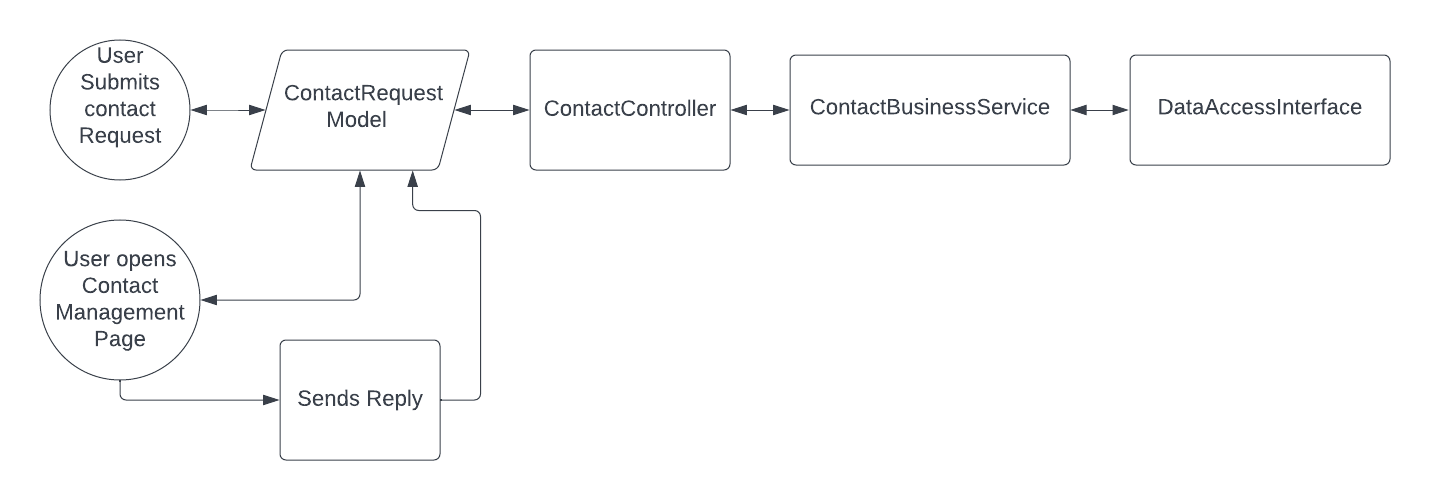
Coming Soon Process Flow



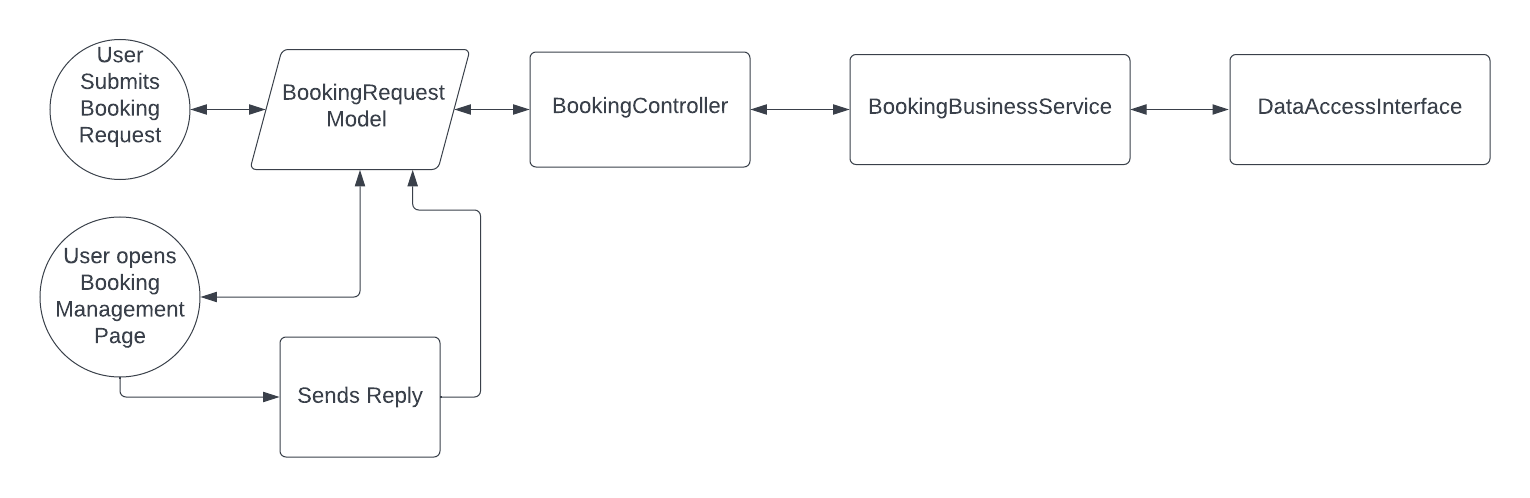
Event Process Flow



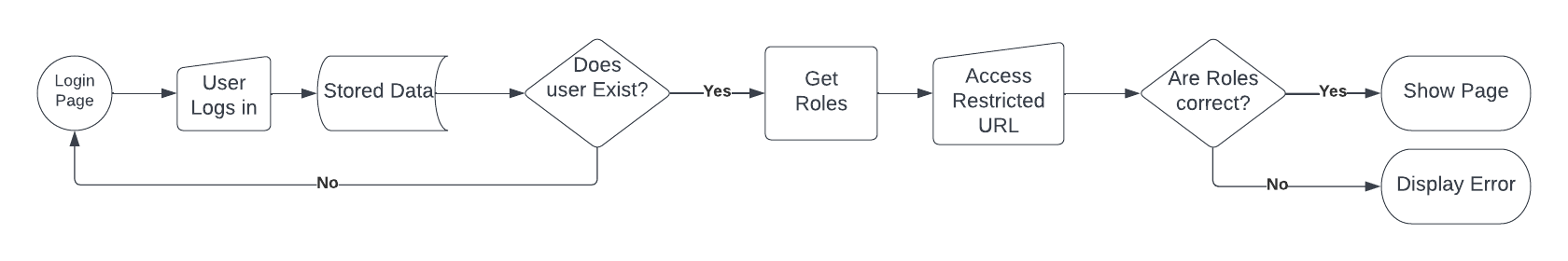
Contact Us Process Flow



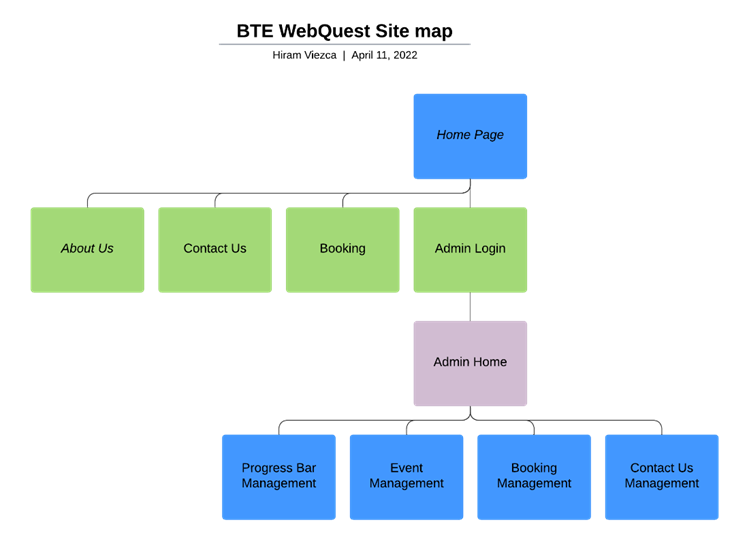
Booking Process Flow



User Process Flow



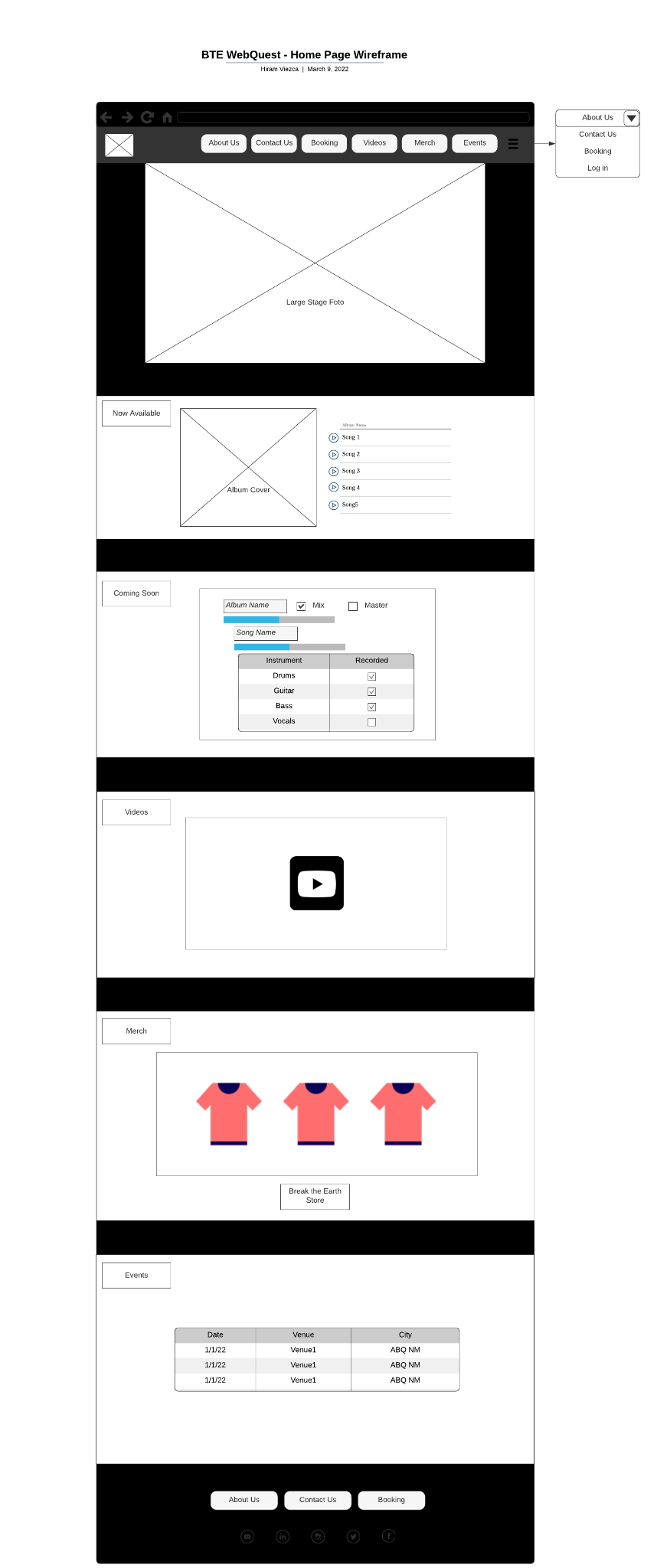
**Sitemap Diagram:**



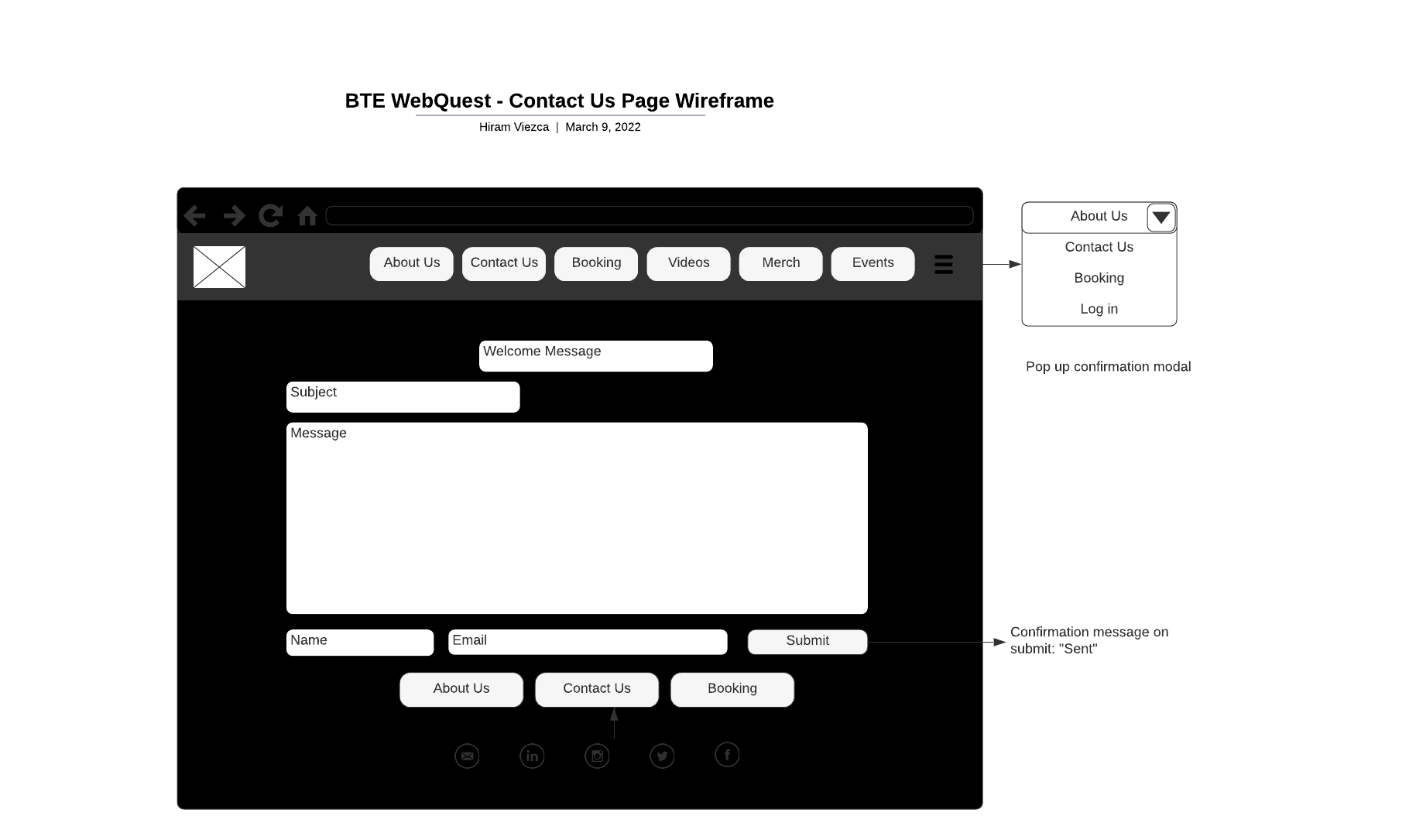
**User Interface Diagrams:**

You should insert any wireframe drawings or white board concepts that were developed to support your application. If you have no supporting documentation, explain the rationale why you are able to leave this section as N/A.

*Main Page*



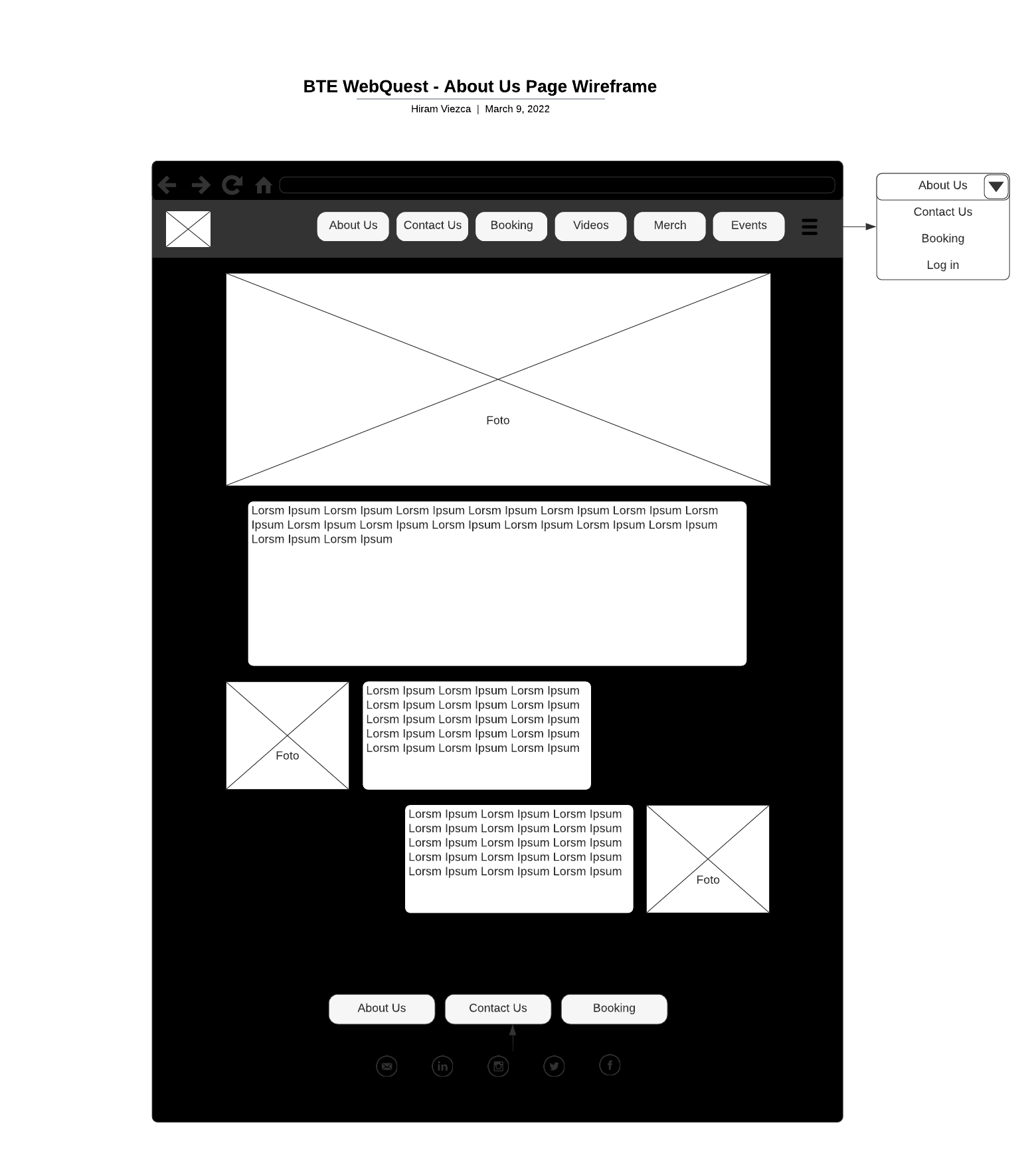
*Contact Us*

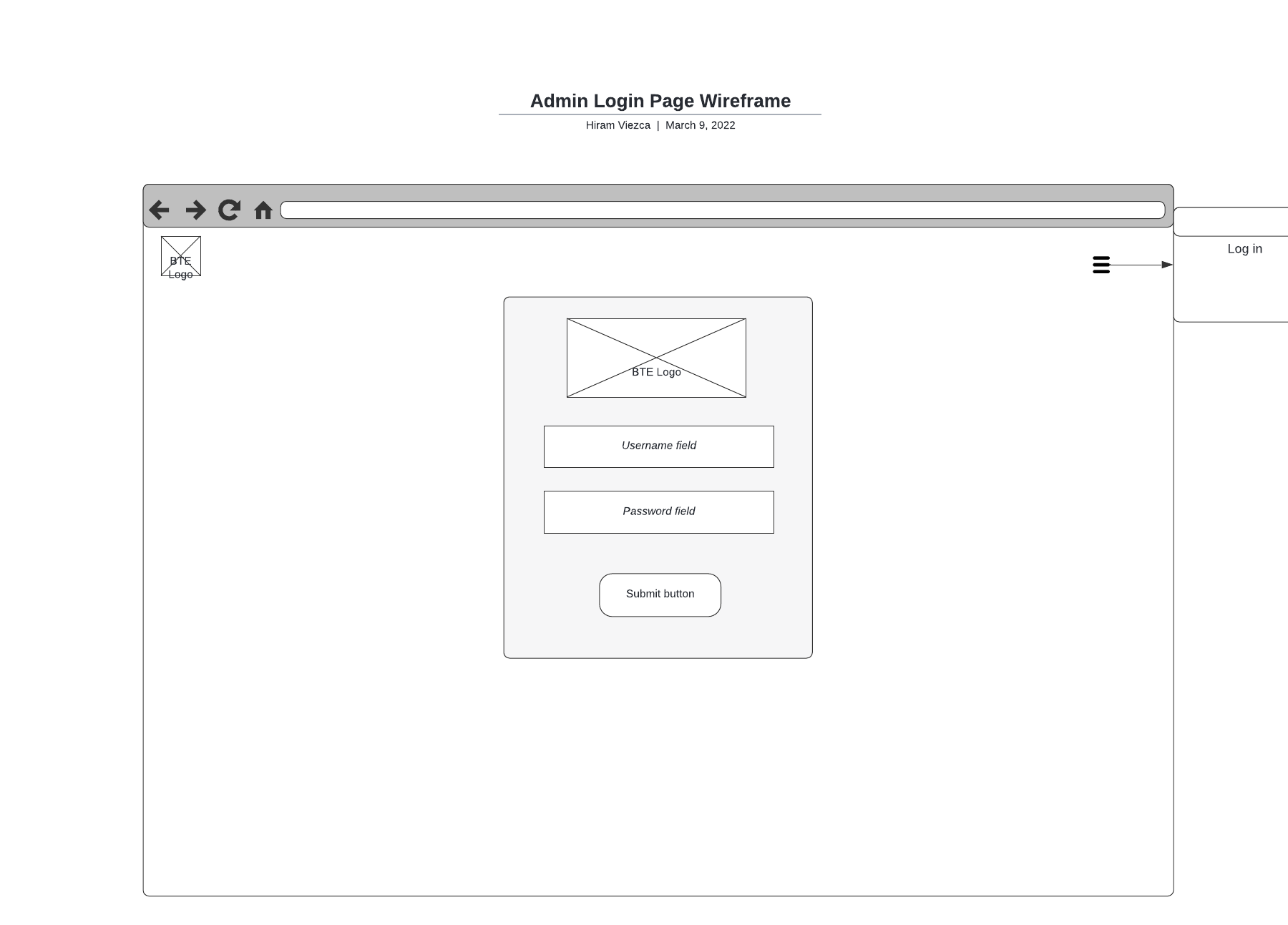


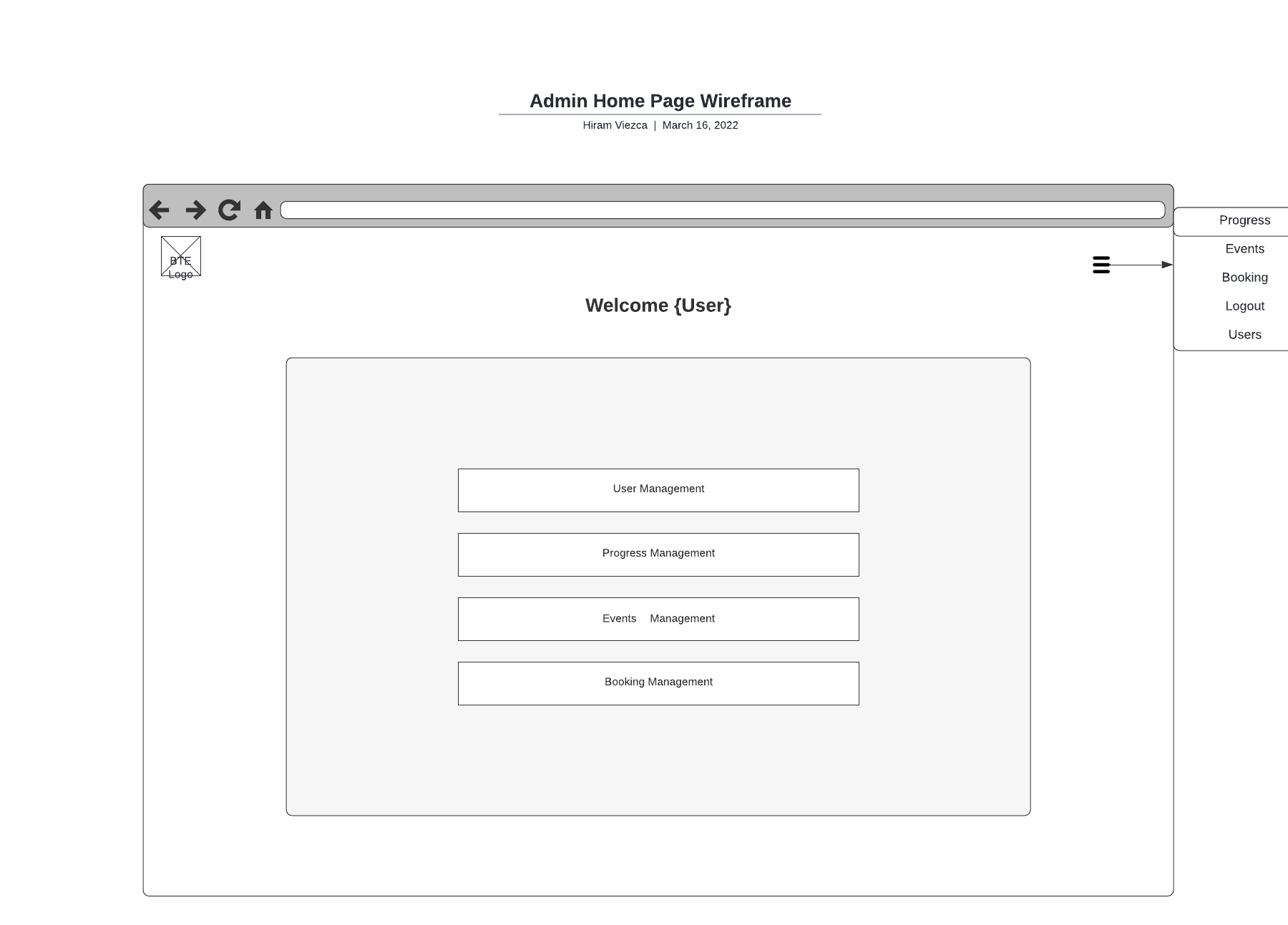
*Book Us*

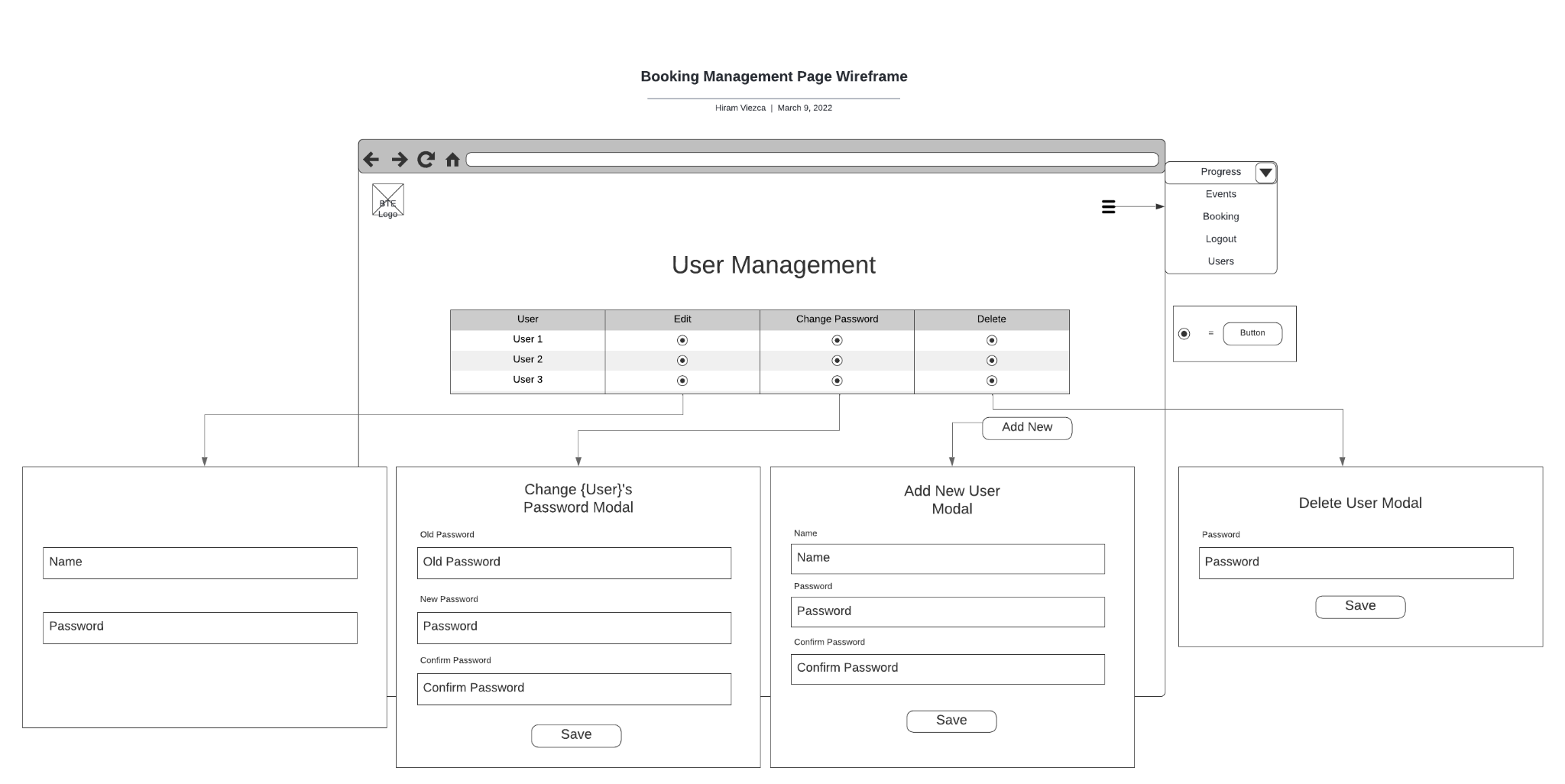


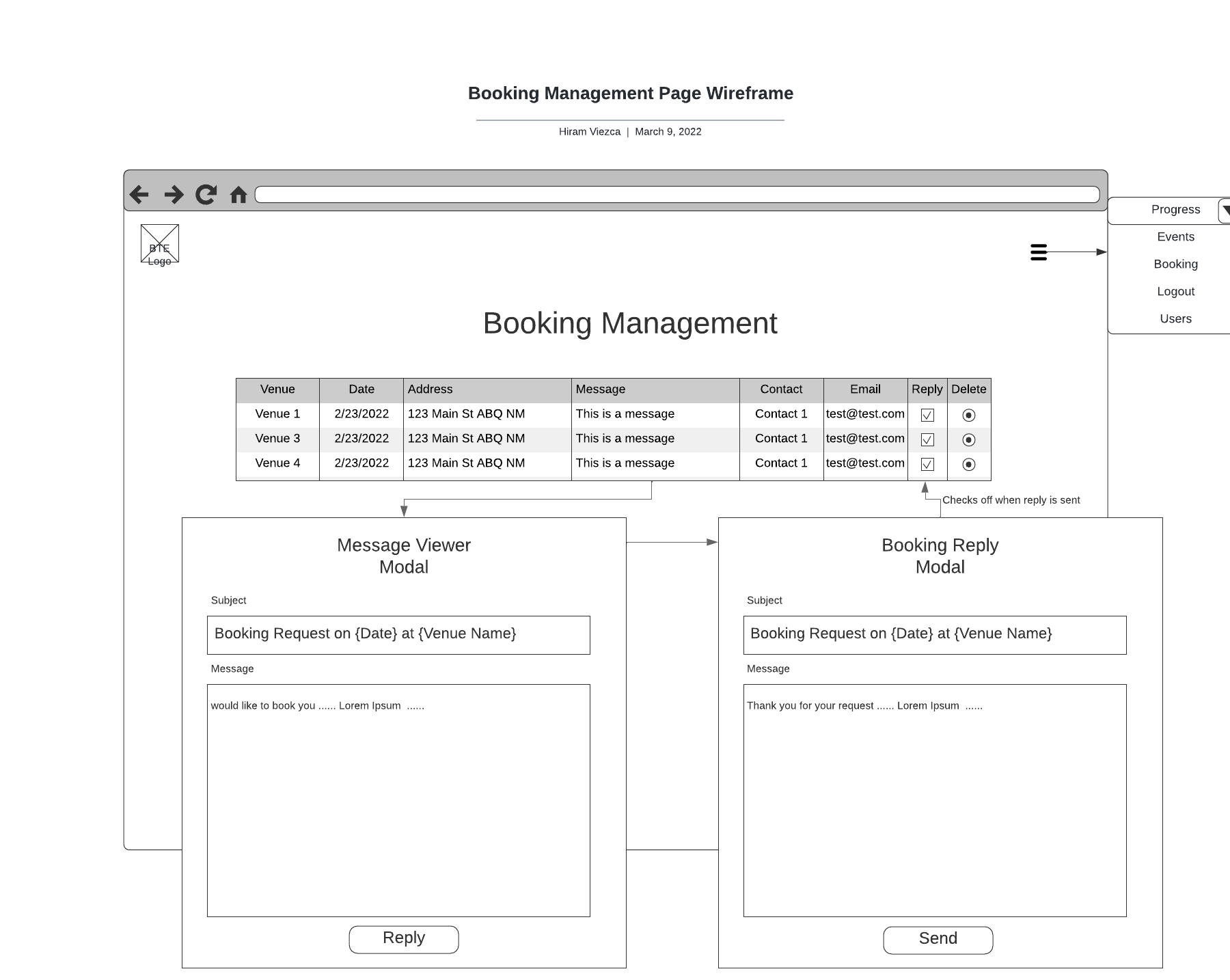
*About Us*

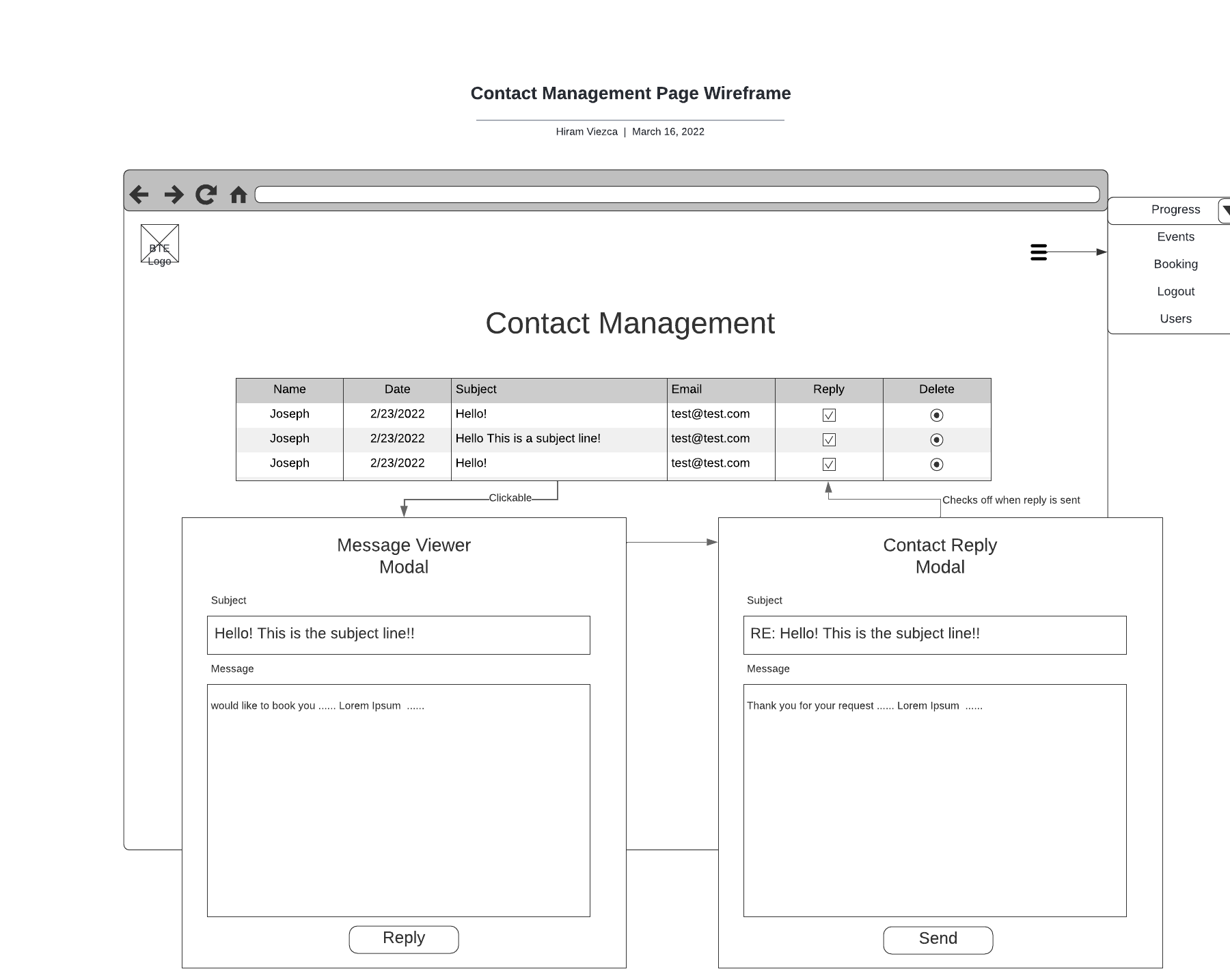
**

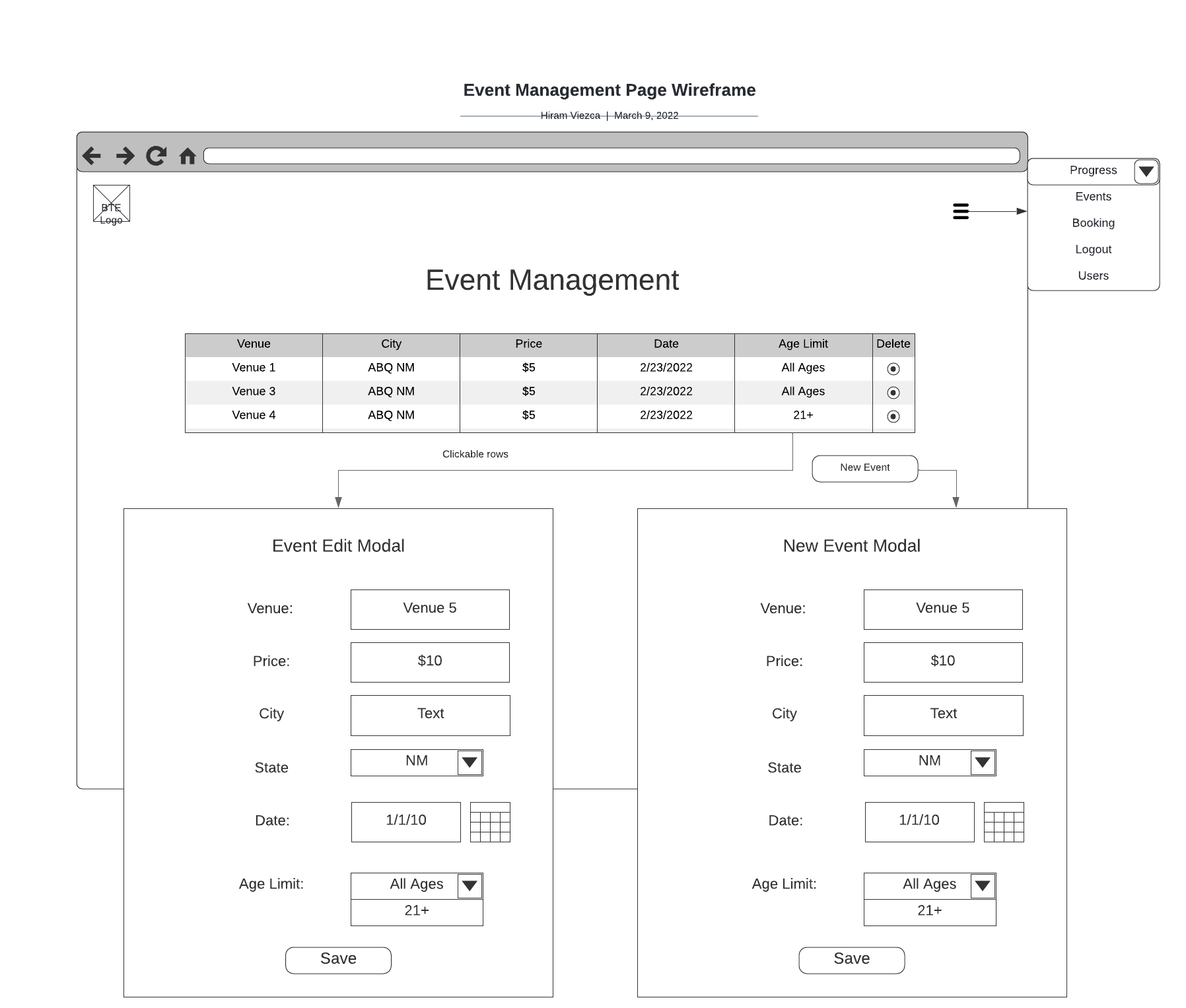
*Admin Login*

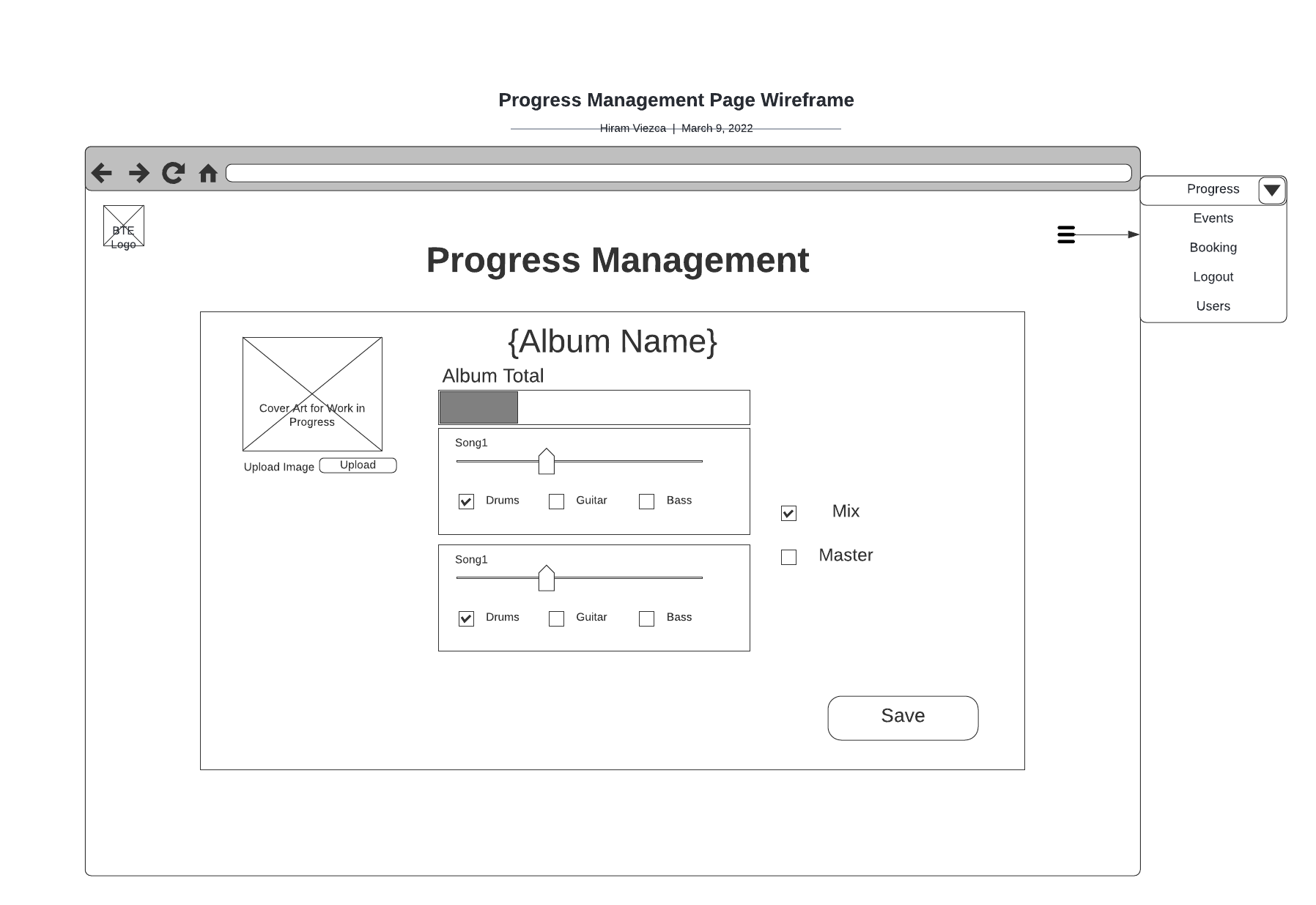
*Admin Home*

*User Management*

*Booking Management*

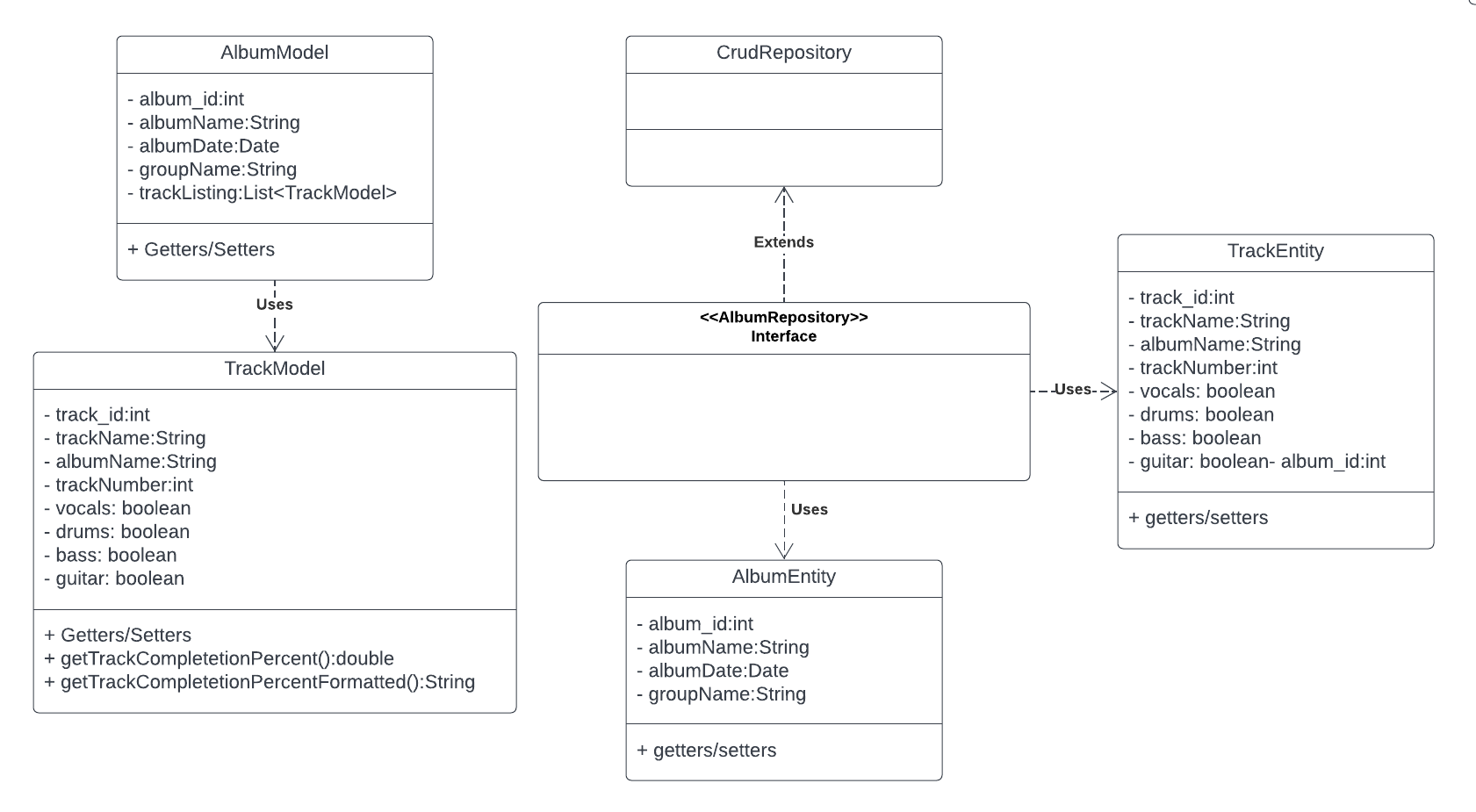
*Contact Us Management*

*Event Management*

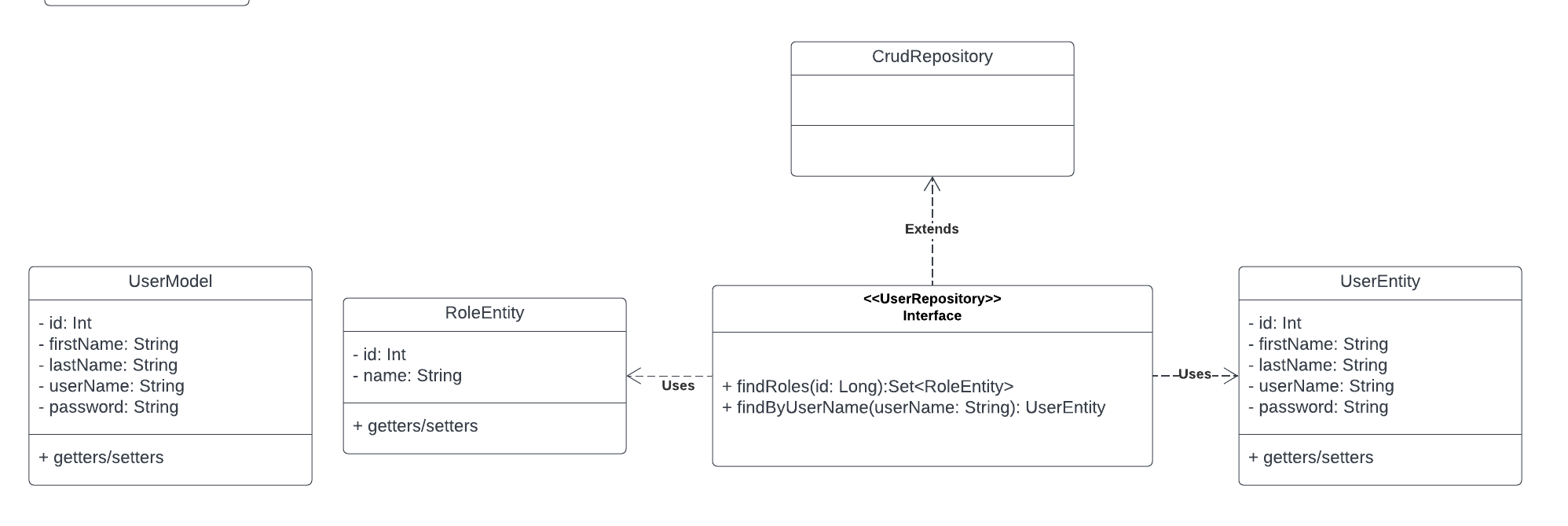
*Progress Management*

**UML Diagrams:**

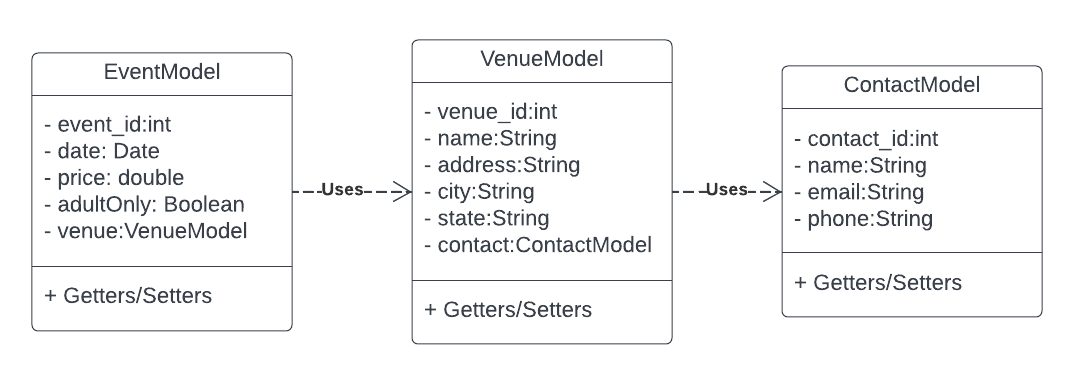
Coming Soon Class Diagram



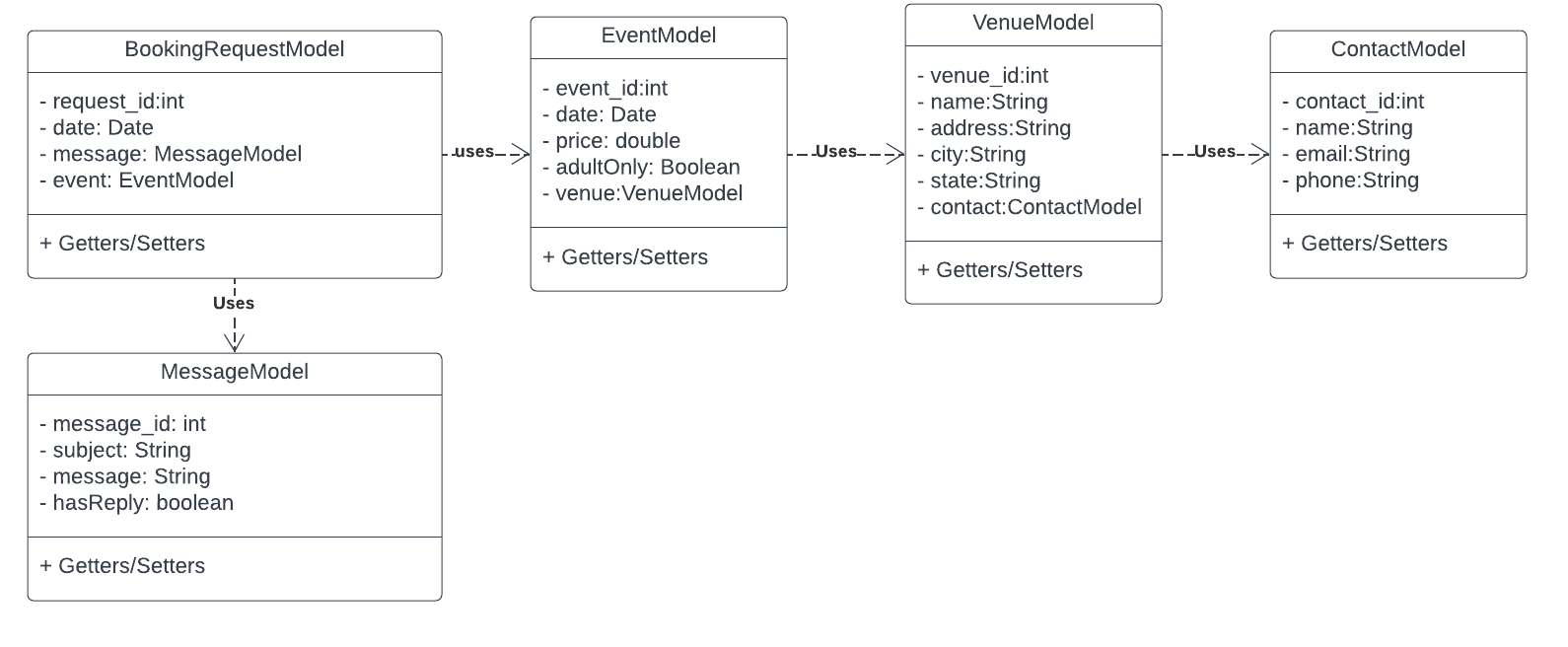
Users Class Diagram



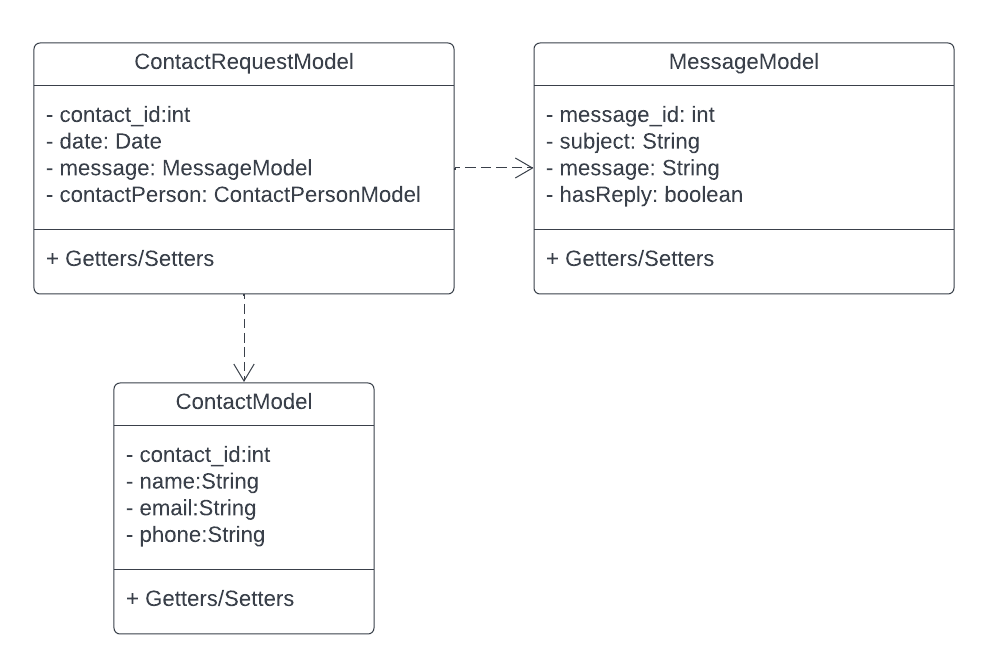
Event Class Diagram



Booking Class Diagram



Contact Class Diagram



**Service API Design:**

This application will not use an API in its current proposed format.

**NFR’s (Security Design, etc.):**

NFR-1

Role management will be handled automatically by the application. The client only wants one ‘Admin’ user for the application. To address this, the DDL script will be set up with an initial user who has ‘Admin’ access. Each additional user created within the application will be assigned the ‘Editor’ role.

NFR-2

Physical security of the application and the cloud server it will be stored on falls upon the cloud service provider chosen by the client. The client will research security and redundancy capabilities of the cloud service providers before selecting one.

NFR-3

Roles are assigned automatically by the application and cannot be updated or assigned by any user preventing them from being able to change theirs.

NFR-4

Form validation will consist of both client side and server-side verification in a combination that ensures security and performance. Client-side verification will consist of ensuring that fields are not null and verifying that passwords are entered correctly twice. Further server-side verification will verify that passwords are correct and user names are unique.

NFR-5

The use of Roles for security in conjunction with the Thymeleaf template engine provides the ability to limit code access to specific URLs and sections of HTML. These security measures will be used to restrict access to administrative pages to only authenticated users. Within the administrative pages specific actions will be limited to those with the ‘Editor’ role as applicable.

NFR-6

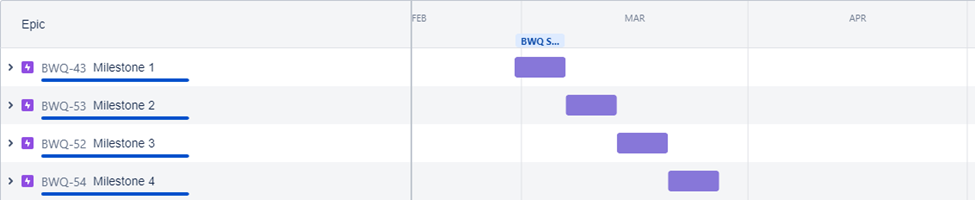
The use of Roles for security in conjunction with the Thymeleaf template engine provides the ability to limit code access to specific URLs and sections of HTML. These security measures will be used to restrict access to administrative pages to only authenticated users. Within the administrative pages all actions will be available to the user with ‘Admin’ role

**Operational Support Design:**

Logging will be implemented with Spring Boots built-in SLF4J logging support. Logging will be conducted for all administrative actions performed by any authorized users.

**Other Documentation:**

Project Schedule:



Programming Schedule



**Appendix A – Technical Issue and Risk Log**

1. Use the template to identify and monitor project issues and risks.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Issues and Risk Log | | | | | | | | |
| **Issue or Risk** | **Description** | **Project Impact** | **Action Plan/Resolution** | **Owner** | **Importance** | **Date Entered** | **Date to Review** | **Date Resolved** |
| I/R | What is the issue or risk? | How will this impact scope, schedule, and cost? | How do you intend to deal with this issue? | Who manages this issue? |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |

**Appendix B – References**

*List all Project Documentation References*

*List all references using APA style*

**Appendix C – External Resources**

|  |  |
| --- | --- |
| **GIT URL:** | *https://github.com/hviezca/BTEWebQuest* |
| **Hosting URL:** | *https://breaktheearth.herokuapp.com* |