# **And X Entertainment** (CSE 490-R)

## Previous Project Information Collection

|  |  |
| --- | --- |
| **Collection Date:**  **Instructor:**  **Student:** | April 29, 2022  Bro. Clements  Guillermo Oliva |

1. **Overview**
   * The customer application is known as “BlitzTix” and its primary objective is to server as a ticket registration system which will manage events per venue and allow the registration of attendees, as well as controlling the transactions and allotment of tickets to available spots for each event and venue.
2. **Requirements (high-level)**
   * **Must have**
     1. App shall support purchasing of tickets
        + Observation: Payment processing integration more likely required.
     2. App shall support data management for:
        + Contact information (users, vendors, and admins)
        + Venue information (seating and location information)
        + Events (tickets, capacity, and seating allocation)
     3. App shall support access levels and secure login
        + Possible access levels include administrator, organizer, venue, and attendee.
        + Each level will be able to modify certain components, such as users, venues, seating, etc.
     4. App shall have its own database store.
        + Currently using PostgreSQL
        + Observation: Need to check that database logic is not hard coded into non-data related tasks.
     5. App shall allow the check-in of attendees.
        + This could include QR codes, electronic tickets, etc. There is no limitation or set method yet.
     6. App shall allow the reporting and analysis of transaction in the system.
        + Observation: This seems to be aimed at data analysis regarding event attendances, revenue, or other business making data. System administration data analysis is not part of the requirements.
     7. App shall connect to third-party applications.
        + Observation: This requirement seemed extremely vague, so it is assumed that it refers integrations to things such as event management and payment processing.
   * **Will/Could have**
     1. App could have a complex security implementation.
        + It is assumed that it was meant that the application could have harden, such as implementing obfuscation, API key revocation policies, etc.
        + This requirement is too vague to be considered at this moment.
     2. App will have the ability to send digital tickets to Apple or Android Wallets.
        + This requirement could be implemented after the user profiles are created.
     3. App will be design for mobile compatibility.
        + This requirement should probably be in the “shall” requirements since the app should be design with mobile first in mind.
     4. App could restrict the use of electronic tickets to certain events.
        + This requirement seems to contradict the “will” requirement of having them in a digital wallet. Is this requirement from the developer perspective?
     5. App could restrict purchase of mass number of tickets to mitigate scalping.
        + There are several mechanisms that could be implemented in this case, but the best why to address it would be by deploying a prototype of the application and analyze the web traffic when tickets are purchased.
        + Machine learning could also be implemented, but this requirement seems to requirement for field testing that unfortunately cannot be done by the developers.
   * **Usability**
     1. Data validation must be present.
        + Ensure back-end data is protected and reduce user frustration.
     2. Visual navigation.
        + This goes with accessibility.
     3. Visual standardization.
        + The UI elements should be well known. E.g., a House icon to symbolize home page.
     4. Minimalistic design.
        + It appears that the customer wants lean visuals.
   * **Performance**
     1. Loading time
        + This is more be also addressed with usability if the team shift to mobile first design.
   * **Security**
     1. Reduce surface exposure.
        + This requirement is too broad as it is, and it is required that the should be a risk analysis.
     2. Authentication
        + Reduce unauthorized access to non-privilege areas of the software.
3. **Some key product details** 
   * 1. Product perspective
        + To be built as a SaaS application.
     2. Product Function
        + State diagrams need to be able to better reflect the user interactions of the system.
     3. Database implementation and status
        + The database relationship structure is already set for the current back-end deployment, this will have to evolve as the integration occurs.
     4. System users
        + Default site behavior for all users is Guest.
        + User registration option available to all users?
          - This registration seems to be aimed at the attendee user. Other users such as the venue or admin, should be added manually or allow any person to register and then they can be added to venues or other privileges?
        + User login after successfully creating an account.
        + User profile available after registration is successful.
        + User can create events if they represent a venue.
     5. Event management
        + All events to be public (when displaying events).
        + Filtering of events based on location.
        + Event reservation capability for the user, which includes choosing the seating location.
        + Purchase confirmation.
        + Payment processing to be handle by third party app.
        + Ticket vending after successful transaction.
     6. Event Registration
        + User can select which event to attend and their seating preference, the user can also accept private event reservations via a code.
        + It is intended that a visual representation of the seating will be provided.
        + The user will be prompted to check out after selecting their seat, a third-party payment application seems to be better suited.
        + The user will be provided with a proof of purchase. They can receive the confirmation via email or other means.
        + The seating allocation will automatically update after each final sale.
     7. Event workflow
        + Promoter
          - Track the start and end of the event.
          - Can check-in 2 hour prior to the event start.
        + Tickets
          - QR codes are the preferred method of check-in
          - Ticket linked to venue, event, time, and seating chose during the purchase step.
          - It is preferred that the ticket is available in electronic format.
          - The QR is the required proof of purchase and will be scanned for validation of purchase.
          - The ticket or event purchases will be available in the history section of the user profile?
        + Monitoring
          - Checked-in to purchased levels can be monitored at any given time by the venue owner and operator.
          - Attendees’ demographic information will be available during and at the end of the event.
          - Venue owners can mark the venue as available after an event ends.
          - A final summary report will be available for each event.
          - A data feed will be provided to be able to analyze all transactions in the application (event related).
     8. Verification

The system verification seems like it needs some work, so only a list of the items to be tested is included in the following list:

* + - * + User sign-up
        + User login
        + User profile
        + Event creation
        + Venue creation
        + Event Reservation
        + Ticket Purchase
        + Ticket Validation (check-in procedure)
        + Event tracking
        + API data access

1. **Final Insights**
   * Front-end: Many of the functionality is hard-coded and it is not ready for integration.
   * Back-end: The use of Docker should be deprecated as this will create unnecessary complexity at the time of integration (must of the team members are unfamiliar with this deployment).
   * Database: Somewhat integrated, but the local deployment seems to be hiding hard-coded modules.