Software Requirements and Design Document

For

Group 1

Version 3.0

Authors:
Andrew Augustine
Anthony Ingle
Tristan Morris
Cole Warren
Dawson Stoller

● Overview (5 points)

We are developing a lottery pooling system where users can buy lottery tickets through our platform and we pool them together by splitting earnings equally amongst tickets bought if at least one of the tickets in the pool wins. The platform will be a website where users can login via Auth0 and buy a certain amount of tickets at a list price. We will then manage keeping track of buyers, buying the tickets and splitting earnings if there is a winning ticket via MariaDB.

When the users login to the website, they will be able to see the different pools that are available for the current ongoing lotteries at that time and will be able to buy one or more lottery tickets via Stripe. They will have the option to input their own number for the ticket as well as view their ticket status.

● Functional Requirements (10 points)

- Implement a system that allows users to join pooling groups with their tickets. This is a high priority, and having the ability for multiple groups is a low priority.
- Allow users to create an account and login as a user to lottery pool service. This is a high priority.
- Have a method for users to pay for a ticket to join the pool. This is a high priority.
- Have a method to calculate the winnings per user after a winning lottery ticket. This is a high priority.
- Allow users to see their tickets purchased through our service. This is a high priority.
- Have a separate login for administrators to print off tickets to be purchased in-person at store locations. This is a high priority.
- Allow users to login online using a standard web browser. This is a high priority.
- Have a method of authentication for users. This is a high priority.

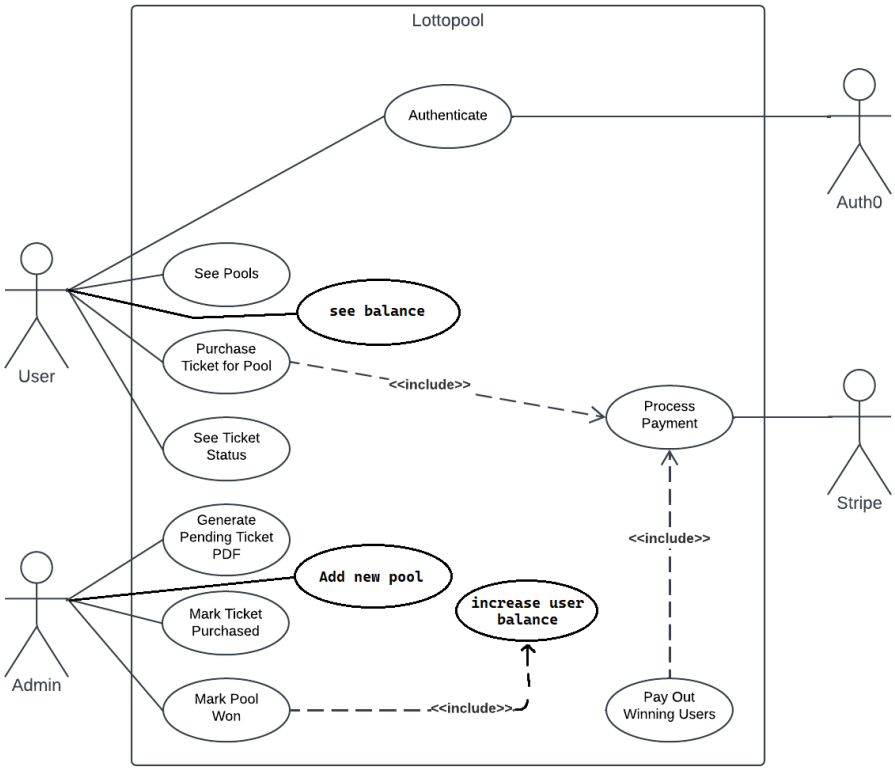
● Non-functional Requirements (10 points)

- Privacy: The system must respect user privacy and protect their personal information, including financial information, from unauthorized access or disclosure.
- Usability: The system must be easy to use and navigate for users of varying technical backgrounds.
- Security: The system must ensure secure authentication and authorization of users and protect against unauthorized access and fraud. This includes using encryption for sensitive data and implementing proper access controls to prevent unauthorized access to the system.
- Reliability: The system must be reliable and maintain data integrity even in the event of system failures or errors.
- Availability: The system must be available 24/7, with minimal downtime or disruption to the users.
- Integration: The system must seamlessly integrate with third-party services like Stripe for payment processing, ensuring secure and efficient transactions.
- Performance: The system must be able to handle users buying tickets simultaneously without slowing down. It must also be able to handle large amounts of data for storing and managing user information and ticket sales.

Note for Sections 4 & 5:

All source images are available in this repo. Just look in the “images” folder

- **Use Case Diagram (10 points)**



- **Class Diagram and/or Sequence Diagrams (15 points)**

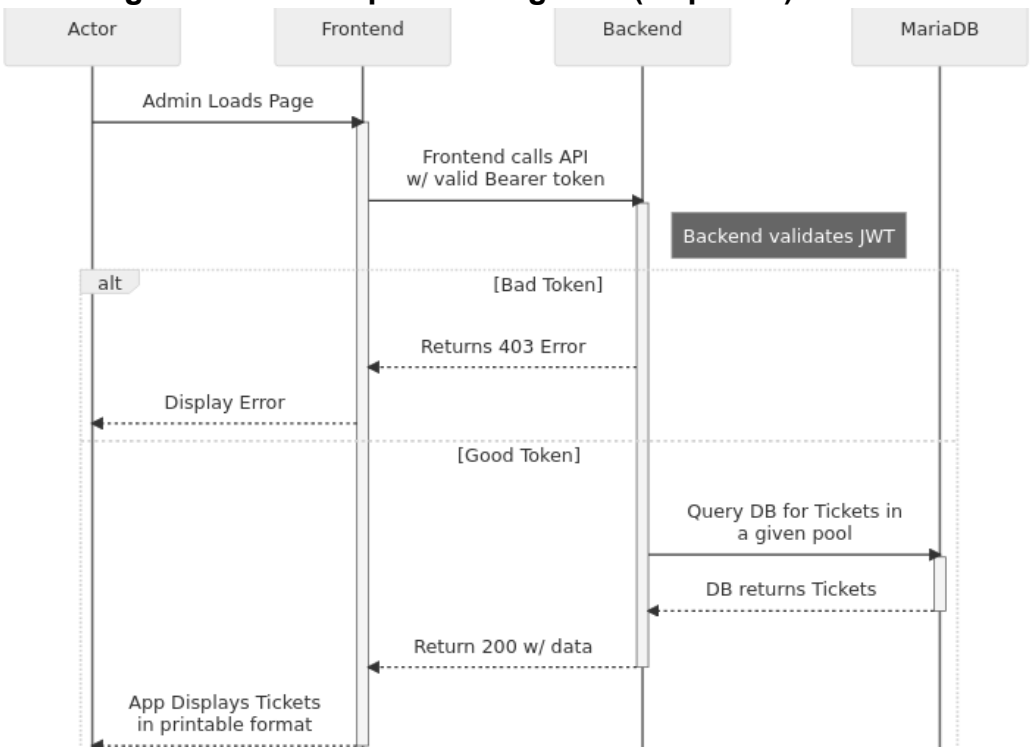


Fig 1. Get unacquired tickets for a given pool

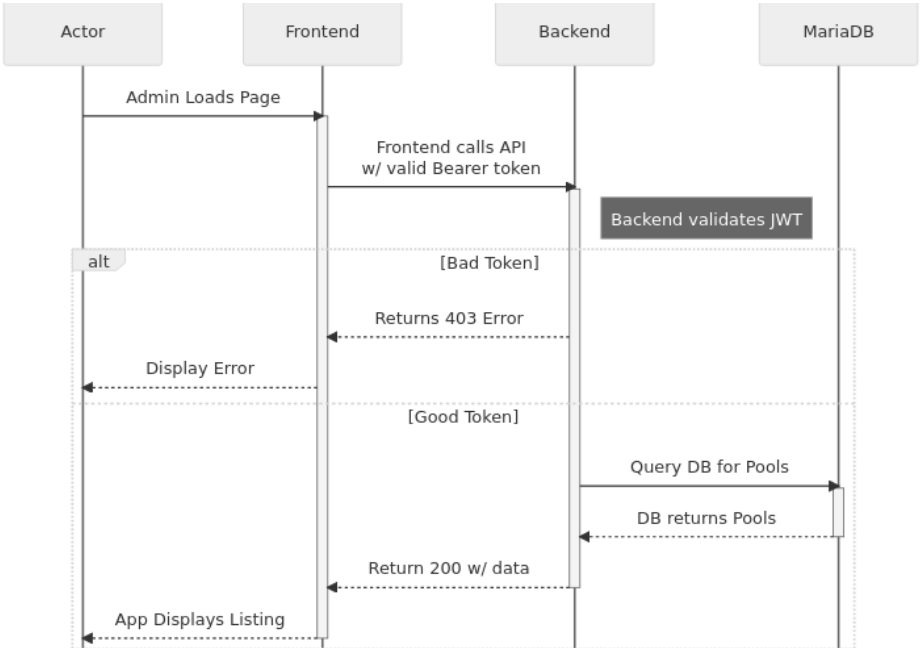


Fig 2. Checking for available pools

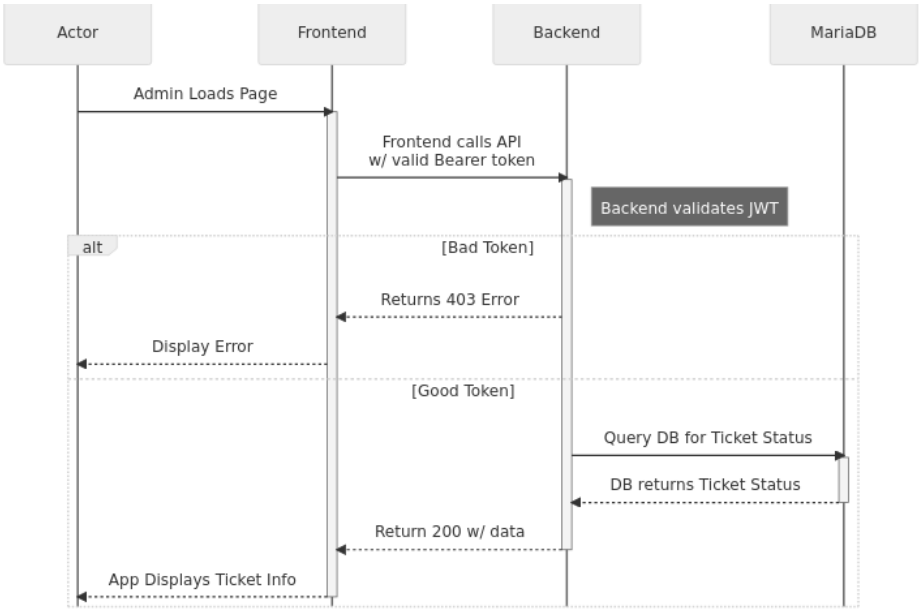


Fig 3. Checking ticket status

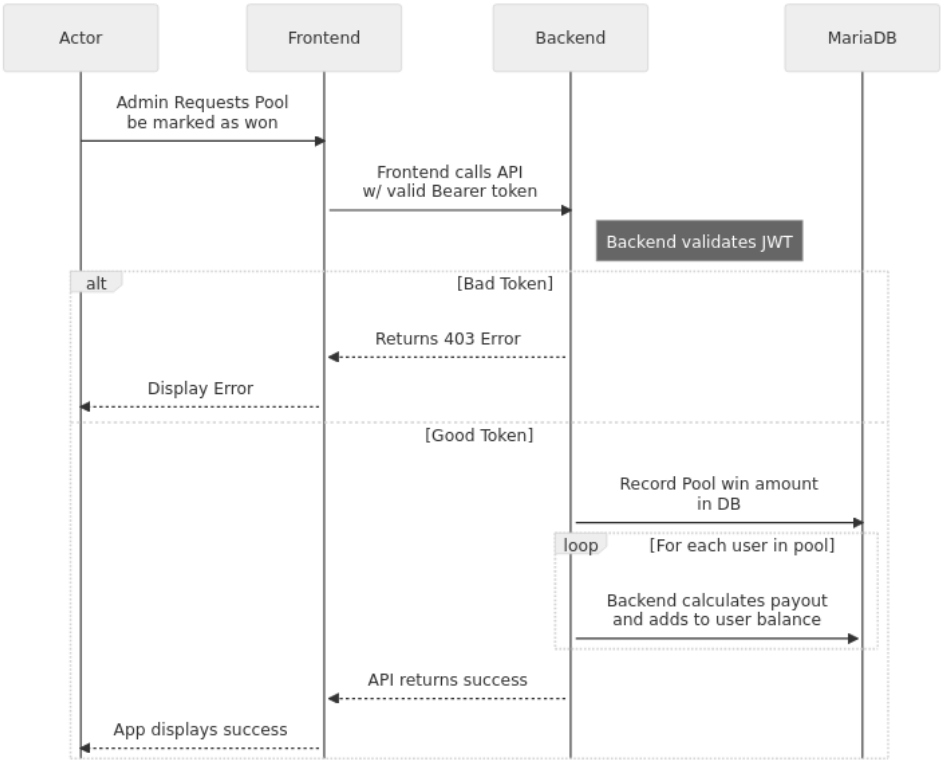


Fig 4. Admin Marking Pool as Won

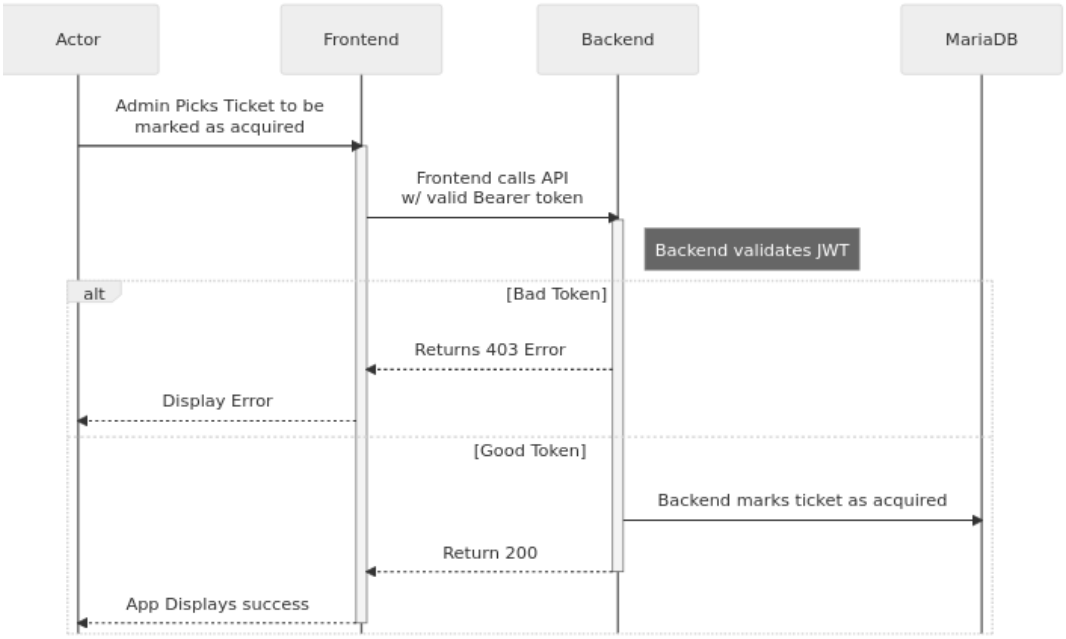


Fig 5. Admin Reporting Ticket Acquisition Procedure

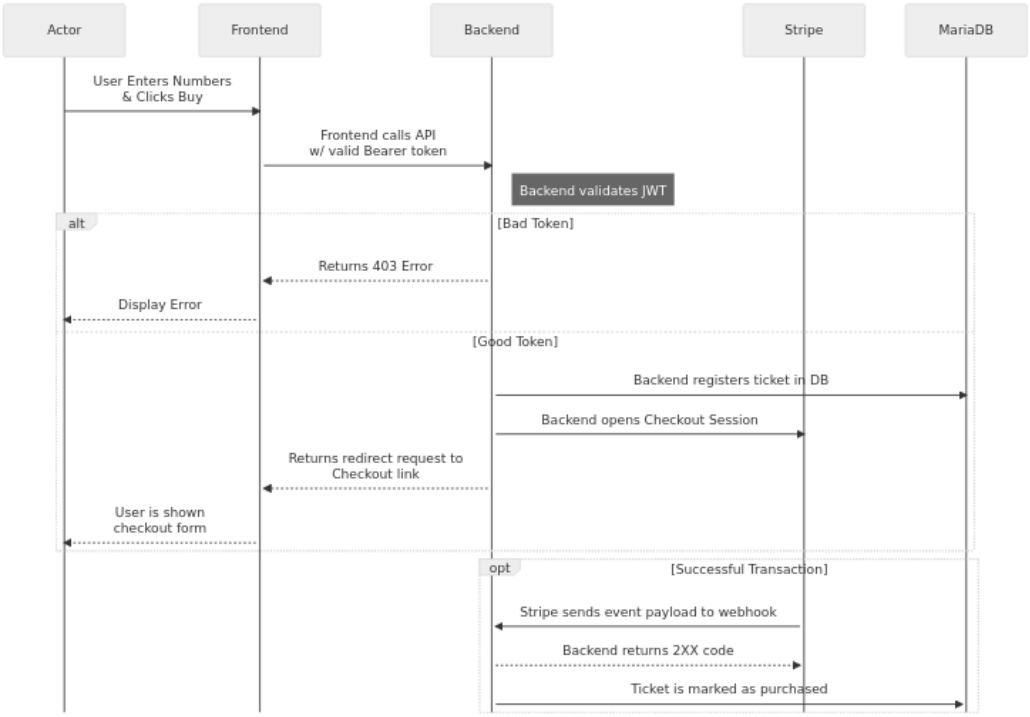


Fig 6. Ticket Purchasing Procedure

- **Operating Environment (5 points)**

Ubuntu 20.04 coexisting with Docker (running debian)

- **Assumptions and Dependencies (5 points)**

We are using a number of third party services that if changed or go offline, our project would be affected. We are using Auth0 for authentication and login to our app, so if this service goes away, no one would be able to login to our app. For payment collection, we are using Stripe. If something happens to Stripe, no one will be able to use the main functionality of our app, which is buying lottery tickets. We are using DigitalOcean for deployment, so if this cloud hosting platform is not available, our app will not be available either. We are using things like Docker, Github Actions and Doppler for continuous deployment through the cloud. If any of these platforms become inaccessible, our automated deployment process is screwed. On top of all these we use a large amount of third party packages and dependencies across both our Python backend and Typescript frontend. All of these services, packages, libraries and dependencies are assumed to be available, functioning and accessible for our application and workflow to be in working condition. We are not reusing any external components for our codebase.