Deliverables 1 - Secure Software Requirement Analysis and Design

A report submitted to the

Singapore Institute of Technology in partial fulfilment of the requirements for the BACHELOR OF INFORMATION AND COMMUNICATIONS TECHNOLOGY (SOFTWARE ENGINEERING / INFORMATION SECURITY)

Team Name: Team 18

Submitted on: 11/06/24

GitHub handle: ICT2101-P11-6

Name	Student Number
Chiu Zheng Hao Jonathan	**2203187**
Goh Yue Jun	**2201471**
Muhammad Fiqri Adam	**2202878**
Nicholas Sng Ray Shiang	**2203197**
Denzel Low	**2202347**
Kee Han Xiang	**2201423**
Jeremy Lim Min En	**2203516**

Table of Contents

Introduction

TicketingHuat is a platform that distributes and manages tickets for various events like concerts and conventions. TicketingHuat also employs a virtual queue system to ensure fair access during high-demand events. The platform also simplifies the booking process with automatic seat assignment, allocating adjacent seats on a first-come, first-served basis. Secure payments are facilitated through Stripe, protecting users' payment information. Users can browse events, purchase tickets, transfer tickets, and manage their bookings through Ticketing huat. Furthermore, TicketingHuat allows ticket transfers within the platform, ensuring authenticity. An administrator page is available for event organisers to manage event details, set ticket prices, and monitor sales.

Stakeholders and Intended Users

The Primary stakeholders and intended users of our platform, Ticketing Huat, would be event attendees and organisers, platform administrators, payment gateway providers, and customer support . To elaborate, event attendees use the platform to browse and purchase tickets for events, while admin use the platform to list events, set ticket prices, and manage events. Platform administrators are responsible for overseeing the operations and maintenance of the platform to ensure it runs smoothly. Lastly, payment gateway providers, such as Stripe, handle secure payment processing and Customer support teams assist users with inquiries and issues.

1 Secure Software Requirement Analysis

1.1 Functional Requirements (FR)

FR	FR Description
Account Management	
FR1	The system shall allow authenticated users to log in to TicketingHuat.
FR2	The system shall allow unauthenticated users to sign up for an account.
FR3	The system shall allow authenticated users to reset their account password.
FR4	The system shall allow authenticated users to log out from their accounts.
Browsing Event Management	
FR5	The system shall allow authenticated and unauthenticated users to search for events.

FR6	The system shall allow authenticated and unauthenticated users to view event details.
Ticketing System	
FR7	The system shall allow authenticated users to enter raffles for different events to bid for seats.
FR8	The system shall allow authenticated users to transfer tickets to other users.
Raffle System	
FR9	The system shall allocate tickets to authenticated users who joined the raffle randomly.
FR10	The system shall notify authenticated users who entered the raffle if they secured seats for the event or not.
Administrator System	
FR11	The system shall allow administrative users to create new events.
FR12	The system shall allow administrative users to edit event details or delete it.
FR13	The system shall allow administrative users to view user accounts, including their tickets and personal information.
Payment System	
FR14	The system shall allow authenticated users to checkout their tickets.
FR15	The system shall allow authenticated users to view their order summary.
FR16	The system shall allow authenticated users to view their ticket receipts.
Customer support	
FR17	The system shall allow users to submit a ticket for customer support.

Table 1. Functional Requirement for TicketingHuat

1.2 Non-Functional Requirement (NFR)

NFR	**NFR Description**
NFR1	The system shall display event search results within 2 seconds under normal operating conditions.
NFR2	The system shall be available to all users at all times, except during maintenance, with a maximum daily downtime of 15 minutes.
NFR3	The system shall have weekly maintenance every Friday night from 0200 hours to 0300 hours.
NFR4	The system shall be accessible via web browsers including Google Chrome, Mozilla Firefox, Safari, and Microsoft Edge, as well as through native mobile apps for iOS and Android.
NFR5	The system must support up to 10,000 users simultaneously accessing the platform during peak times, such as major event releases or sales.
NFR6	The system shall respond to user requests within 10 seconds.
NFR7	The system shall provide authenticated users with 1 hour period to finalize their purchase upon notification before the tickets are voided.
NFR8	The system shall only allow authenticated users to enter 1 raffle entry per event.
NFR9	The system shall send an email to users to reset their password within 5 minutes.
NFR10	The system shall send OTP to verify the user's phone number upon registration.

Table 2. Non Functional Requirement for TicketingHuat

1.3 Security Requirements

1.3.1 Secure Functional Requirements

FR	FR Description	SFR	SFR Description
FR1	The system shall allow authenticated users to log in to TicketingHuat.	SFR1	The system shall not allow brute force, dictionary attack, MiTM, session hijacking, credential stuffing and SQLi attacks.
FR2	The system shall allow unauthenticated users to sign up for an account.	SFR2	The system shall not allow account registration using duplicate email addresses.
			The system shall not allow the creation of an user account not following the strong password policy.
			The system shall not allow SQLi attacks.
FR3	The system shall allow authenticated users to reset their account password.	SFR3	The system shall not allow a user to reset their account password without a valid reset token.
			The system shall not allow SQLi and XSS attacks.
FR4	The system shall allow authenticated users to log out from their accounts.	SFR4	The system shall not allow a user to log out without ensuring that all client and server-sided sessions are cleared. A new session identifier is generated to prevent session hijacking and CSRF.
FR5	The system shall allow authenticated and unauthenticated users to search for events.	SFR5	The system shall not allow the event browsing functionality to be vulnerable to SQLi, XSS attack.
FR6	The system shall allow authenticated and unauthenticated users to view event details.	-	-

FR7	The system shall allow authenticated users to enter raffles for different events to bid for seats.	SFR6	The system shall not allow duplicate entries of raffle entries.
			The system shall not allow DoS attacks.
FR8	The system shall allow authenticated users to transfer tickets to other users.	SFR7	The system shall not allow the ticket transfer functionality to be vulnerable to CSRF, and MiTM attacks.
FR9	The system shall allocate tickets to authenticated users who joined the raffle randomly.	SFR8	The system shall not allow the raffle allocation to be vulnerable to cryptographic failures.
			The system shall not allow administrator users to manipulate the ticket allocation in the database.
FR10	The system shall notify authenticated users who entered the raffle if they secured seats for the event or not.	-	-
FR11	The system shall allow administrative users to create new events.	SFR9	The system shall not allow unauthorised access, privilege escalation, session hijacking and data theft.
FR12	The system shall allow administrative users to edit event details or cancel them.	SFR10	The system shall not allow unauthorised access, privilege escalation, session hijacking and data theft.
FR13	The system shall allow administrative users to view user accounts, including their tickets and personal information.	SFR11	The system shall not allow unauthorised access, privilege escalation, session hijacking and data theft.
FR14	The system shall allow authenticated users to check their tickets.	SFR13	The system shall not allow XSS, SQLi, brute force attacks.
FR15	The system shall allow authenticated users to view their order summary.	SFR14	-
FR16	The system shall allow authenticated users to view their ticket receipts.	SFR15	The system shall not allow this functionality to be vulnerable to information disclosure.

FR17	The system shall allow users to submit a ticket for customer support.	SFR16	The system shall not allow XSS attack.

Table 3. Secure Functional Requirements

1.3.2 Functional Security Requirements

1.3.2.1 Confidentiality

This table details the Functional Security Requirements (FSRs) implemented to safeguard the confidentiality of user data within the system. These FSRs encompass various security measures to achieve this objective.

FSR	**FSR Description**
FSR1	The system shall mask the password field, hash and salt passwords before storing them in the database, with a minimum password length of 8 characters and maximum of 12 characters.
FSR2	The system shall use HTTPS for communications between client and server.
FSR3	The system shall only allow authenticated users to view their own account details.

Table 4. Confidentiality Functional Security Requirements

1.3.2.2 Integrity

This table details the Functional Security Requirements (FSRs) implemented to safeguard the integrity of data and system components within the application. These FSRs encompass various security measures to ensure data accuracy, code integrity, and secure access control.

FSR	**FSR Description**
FSR4	The system shall automatically assign a unique identifier to each ticket and generate unique order reference numbers upon successful ticket booking
FSR5	The system shall integrate with Stripe to verify the integrity of payment before, during, and after the transaction.
FSR6	The system shall verify software installations or updates coming from the expected source, using the digital signatures they provide.

Table 5. Integrity Functional Security Requirements

1.3.2.3 Availability

This table details the requirements implemented to ensure the availability of system components within the application.

FSR	**FSR Description**
FSR7	The system shall ensure high availability with an uptime of 99%, excluding the maintenance period.
FSR8	The system shall support up to 10,000 users concurrently accessing and using the system at any given time.
FSR9	The system shall restore critical features (ticket purchase, administrator management, and queueing system) to normal operations within 1 hour of any disruption.
FSR10	The system shall implement a CDN, and DNS load balancing to speed up page load time.

Table 6. Availability Functional Security Requirements

1.3.2.4 Authentication

This table details the requirements implemented to ensure there is authentication prior to the access of system components within the application.

FSR	**FSR description**
FSR11	The system shall authenticate users using their email and password, and support MFA for all user accounts
FSR12	The system shall deploy access control mechanisms, such as MFA to prevent access to unauthorized areas of the website.
FSR13	The system shall prompt all users to change their passwords every 90 days since their last password modification date.

Table 7. Authentication Functional Security Requirements

1.3.2.5 Authorization

This table details the requirements implemented to ensure there is authorization prior to the access of certain system components within the application.

FSR	FSR description
FSR14	The system shall deny by default except for public resources.
FSR15	The system shall generate a session once the user has authenticated themselves, and will not be prompted for their credentials until their session ends.
FSR16	The system shall give unauthenticated users read-only permission, as part of the guest user role, allowing users to search and view event details without logging in.
FSR17	The system shall give authenticated users read and write permission, as part of the general user role. This includes the functionality to search for events, purchase tickets, view order summaries and receipts, transfer tickets, and update account details.
FSR18	Creating, updating, and deleting events, as well as viewing and deactivating user accounts, and administrator UI will be restricted to the administrator role.

Table 8. Authorization Functional Security Requirements

1.3.2.6 Accountability

This table details the requirements implemented to ensure the accountability of system components within the application.

FSR	FSR description
FSR19	The system shall log activities such as (user actions, payment, administrative actions and failed login attempts) and specify the details captured such as the (email, action type, timestamp, and IP address where the requests originated from).
FSR20	The system shall store all log entries in a write-once, read-many (WORM) format to prevent tampering and restrict log access to authorized personnel only, using role-based access control.

Table 9. Accountability Functional Security Requirements

1.3.3 General Security Requirements

1.3.3.1 Session Management

This table details the requirements implemented to ensure there is appropriate session management within the application.

FSR	FSR description
FSR21	The system shall issue a unique and random session identifier (session ID) that is not guessable.
FSR22	The system shall not allow concurrent logins from multiple devices or browsers. If a new session is initiated, any existing sessions associated with the same user account shall be automatically terminated.
FSR23	User sessions shall automatically expire after 30 minutes of inactivity. Users shall be alerted with a warning message 3 minutes before automatic logout, providing them with the option to extend their session.
FSR24	The system shall provide users with the ability to manually terminate their session via a “Logout” function. Upon logout, the session ID must be immediately invalidated and cannot be reused.

Table 10. Session Management Functional Security Requirements

1.3.3.2 Error Management

This table details the requirements implemented to ensure there is error management and handling within the application.

FSR	FSR description
FSR25	The system displays generic error messages to users without revealing any sensitive information including personal information, and about the underlying architecture, database schema, or code.
FSR26	The system shall catch errors using both client-side and server-side validations.
FSR27	The system shall catch and handle unexpected errors. Errors shall provide detailed error information to administrator users which includes user actions leading up to the error.

Table 11. Error Management Functional Security Requirements

1.3.3.3 Configuration Management

This table details the requirements implemented to ensure secure configuration management within the application.

FSR	FSR Description
FSR28	The system shall ensure all sensitive data are stored in an env file.
FSR29	The system shall initialize all configurations upon startup or configuration changes by applying predefined security settings.
FSR30	The system shall log all access to configuration settings and changes made by capturing administrator details.

Table 12. Configuration Management Functional Security Requirements

1.3.3.4 Code Management

This table details the requirements implemented to ensure secure code management in the application.

FSR	FSR Description
FSR31	The system shall store all source code in a private GitHub repository with access restricted to authorized personnel only.
FSR32	The system shall implement branch protection rules on the main and release branches, requiring pull request reviews, status checks before merging and prohibiting direct commits.
FSR33	The system shall integrate with GitHub advanced security features, including code scanning to automatically detect security vulnerabilities and coding errors in the pull request and main branches.
FSR34	The system shall utilize GitHub actions to automate the build, test, and deployment pipeline and integration test on all commits.

Table 13. Code Management Functional Security Requirements

1.3.4 Non-Functional Security Requirements

This table details the requirements implemented to ensure operational security in the application

NFSR	NFSR Description
NFSR1	The system shall use 32-bit salt together with PBKDF2 with an iteration count of 100000.
NFSR2	The system shall ensure logs are reviewed at least once a week to detect and alert on potential security or misuse incidents.
NFSR4	The system shall encrypt all personal data using AES-256 and in transit using TLS, complying with the CSA's recommendations.
NFSR5	The system shall have a comprehensive incident response plan that includes immediate notification procedures compliant with the PDPA's requirements for data breaches.
NFSR6	The system shall ensure all user agreements and privacy policies are compliant with the Personal Data Protection Act (PDPA) and communicated to users.
NFSR7	The system shall be equipped with redundancy plans and disaster recovery procedures to ensure system availability and data integrity in case of hardware failure or cyber-attacks.
NFSR8	The system shall undergo regular security assessments during weekly maintenance, including penetration testing and vulnerability scans, by certified professionals to identify and mitigate potential security weaknesses.
NFSR9	A ready-to-deploy image of the virtual machine (VM) and a script is prepared to ensure quick server setup in case of failure.
NFSR10	The system shall ensure audit trails are accessible for inspection by authorized internal auditors or regulatory bodies without affecting the integrity of the data.
NSFR11	The system shall lock an account for 5 minutes after 5 failed attempts in inputting payment PIN and failed login.
NSFR12	Logs shall be rotated every 3 months.

Table 14. Non-Functional Security Requirements

1.3.5 Secure Development Requirements

This table details the requirements implemented to ensure secure development practices in the application

SDR	**SDR Description**
SDR1	The code repository shall not store any sensitive information, such as API keys.
SDR2	The system shall adhere to secure coding practices provided by OWASP.
SDR3	The system shall do manual code review, automated testing, and static and dynamic analysis at every step of the development lifecycle.
SDR4	Code merges into the git main branch shall be approved only after a code review by another developer.
SDR5	The system shall conduct real-time monitoring and logging to detect suspicious activities at every step of the development lifecycle.

Table 15. Secure Development Requirements

2 Secure Software Design

2.1 Use Case, Abuse/Misuse Case Diagrams

This section describes the use case and the potential misuse case of the TicketingHuat system.

2.1.1.1 Use case

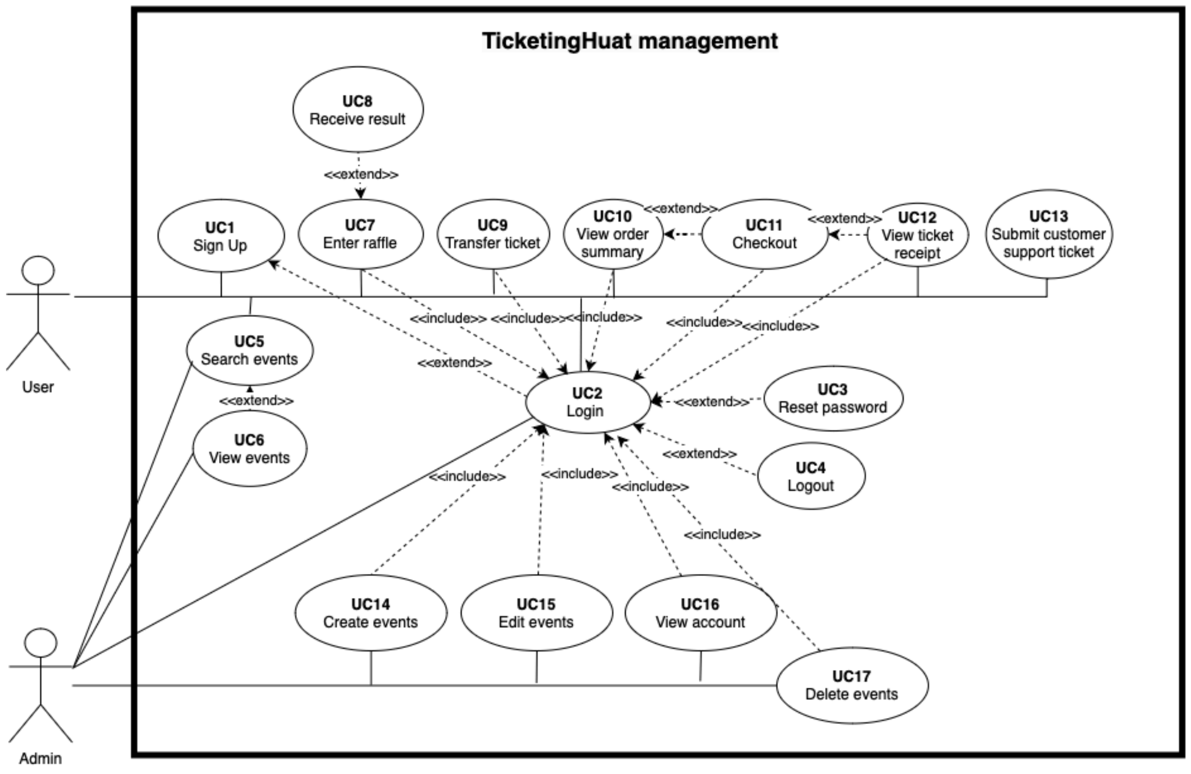

Figure 1. Use Case Diagram on TicketingHuat Management

UC	**UC Description**
UC1: Sign up	Allow unauthenticated users to sign up as users using email address and password.
UC2: Login	Allow authenticated user to log into the system.
UC3: Reset Password	Allow authenticated users to reset their password.
UC4: Logout	Allow authenticated users who are logged into the system to logout of their account.
UC5: Search Events	Allow all users to search for events.

UC6: View Events	Allow all users to view event details.
UC7: Enter Raffle	Allow authenticated users to enter a raffle.
UC8: Receive Result	Allow authenticated users to receive raffle results.
UC9: Transfer tickets	Allow authenticated users to transfer tickets to another authenticated user.
UC10: View order summary	Allow authenticated users to view event and ticket details before purchasing tickets.
UC11: Checkout	Allow authenticated users to make payment for the tickets they won from the raffle.
UC12: View ticket receipt	Allow authenticated users to view ticket receipt upon successful ticket purchase.
UC13: Submit Customer Support ticket	Allow authenticated users to submit support tickets if they have issues or inquiries.
UC14: Create Event	Allow administrator to create a new event.
UC15: Edit Event	Allow administrator to edit event details.
UC16: View Account	Allow administrator to view user account details, including their tickets and basic information.
UC17: Delete events	Allow administrator to delete existing events from the system.

Table 16. Use Case Description

2.1.1.2 Misuse Case

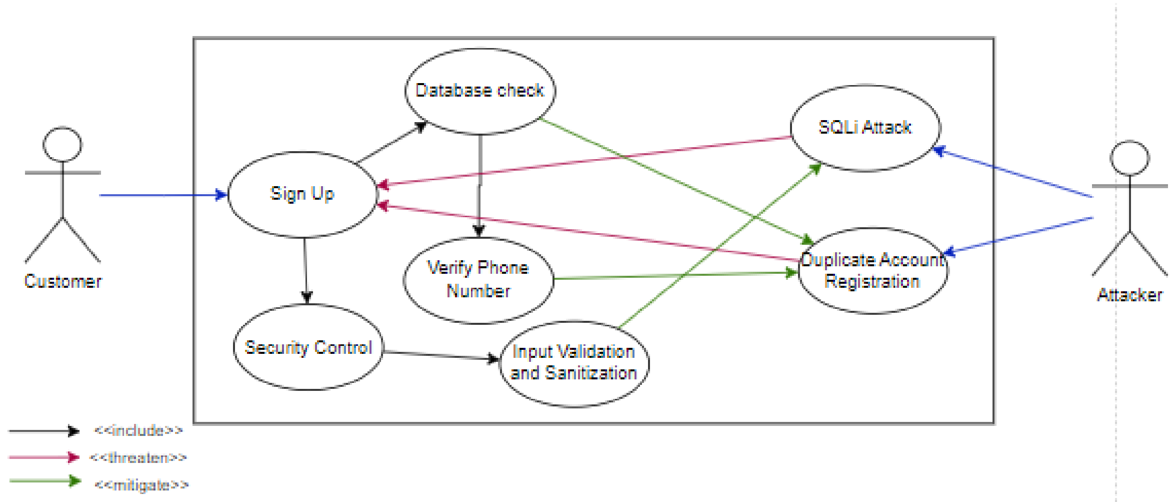

Figure 2. MC1: Sign Up

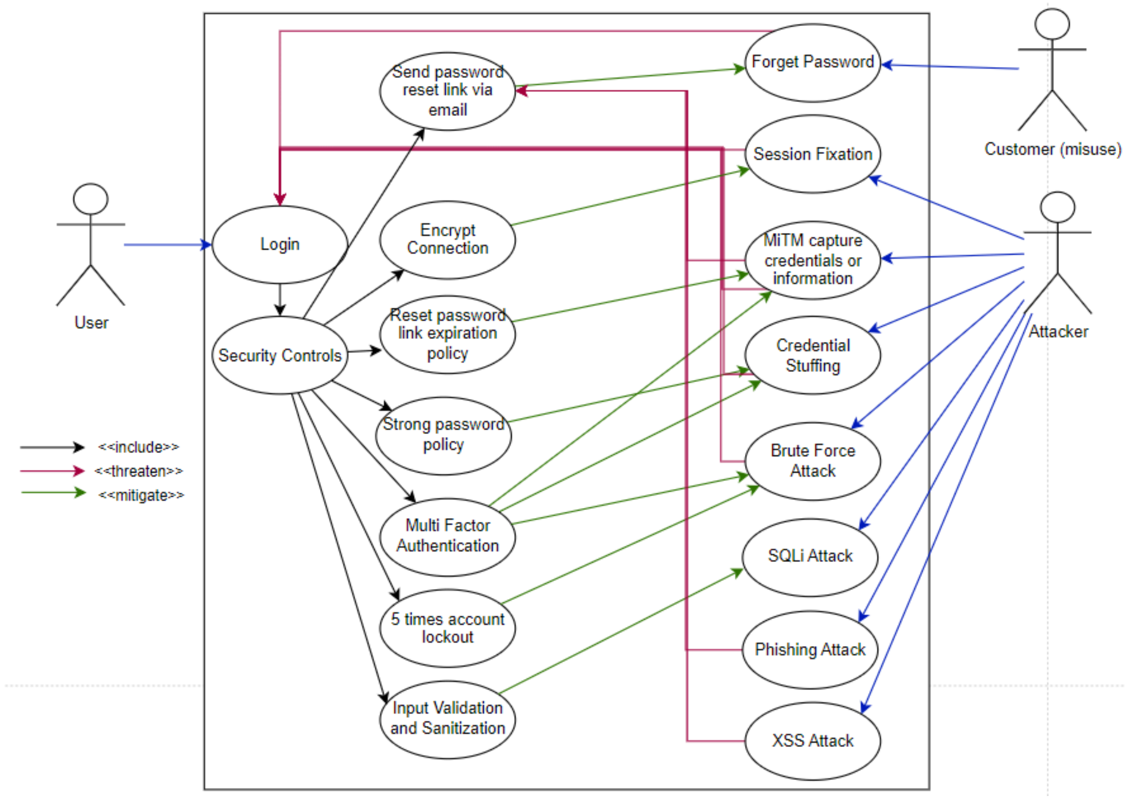

Figure 3. MC2: Login

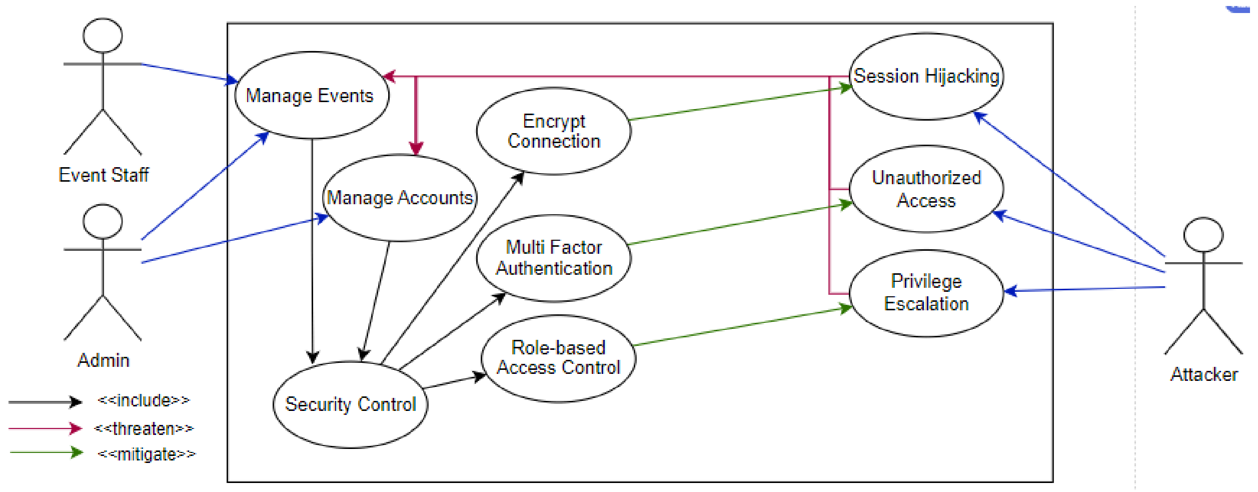

Figure 4. MC3: Administrator Functionality

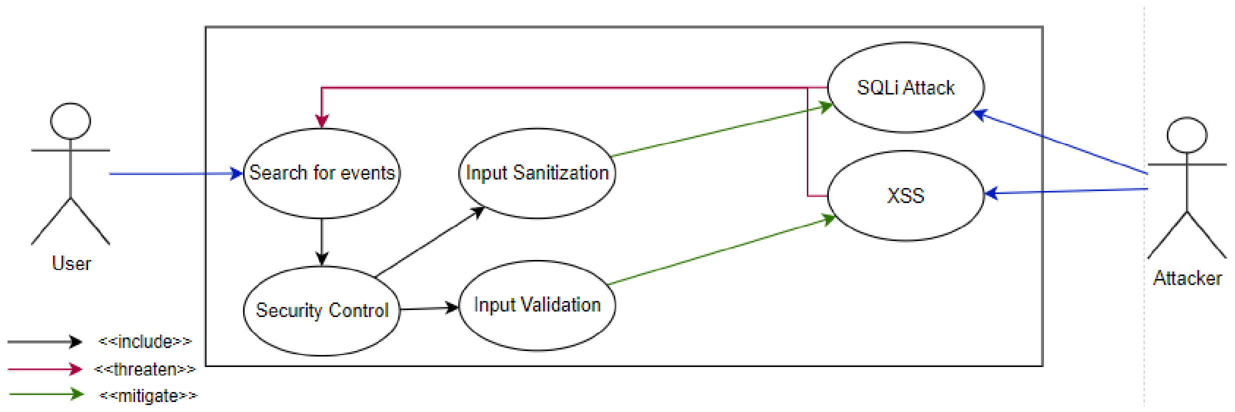

Figure 5. MC4: Search Event

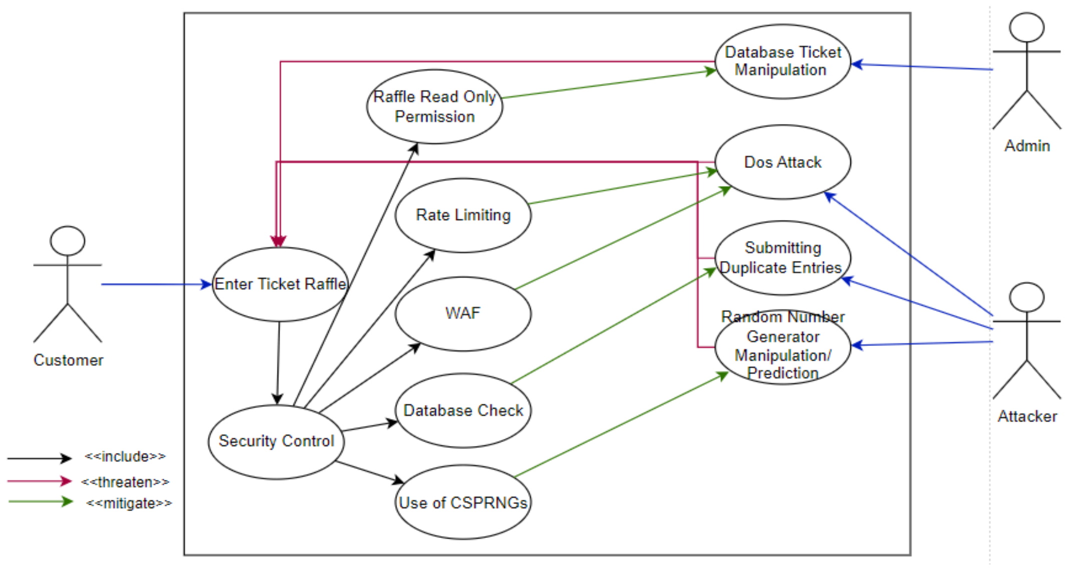

Figure 6. MC5: Raffle Ticket

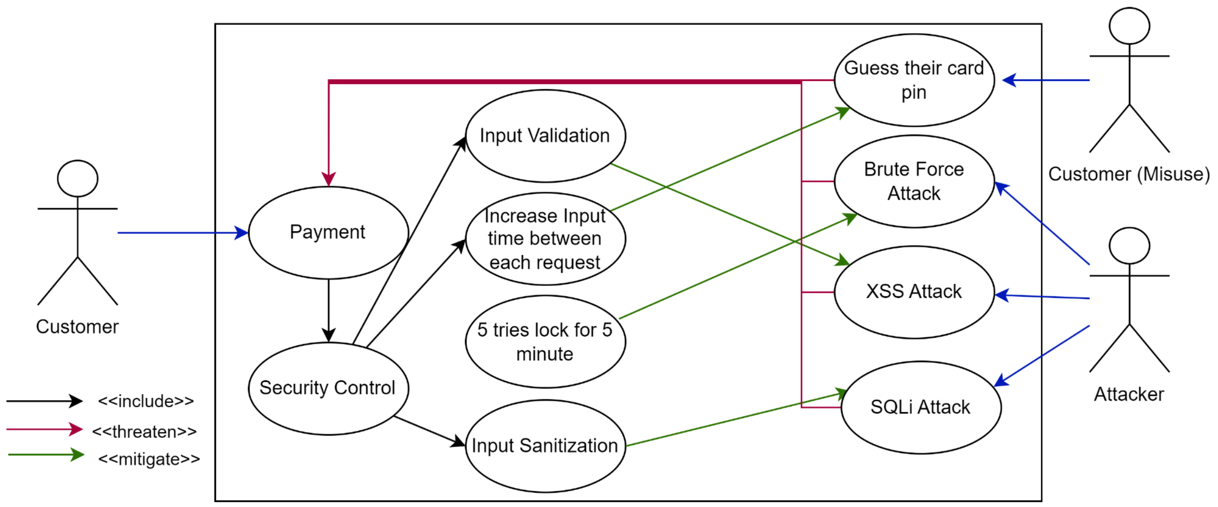

Figure 7. MC6: Payment

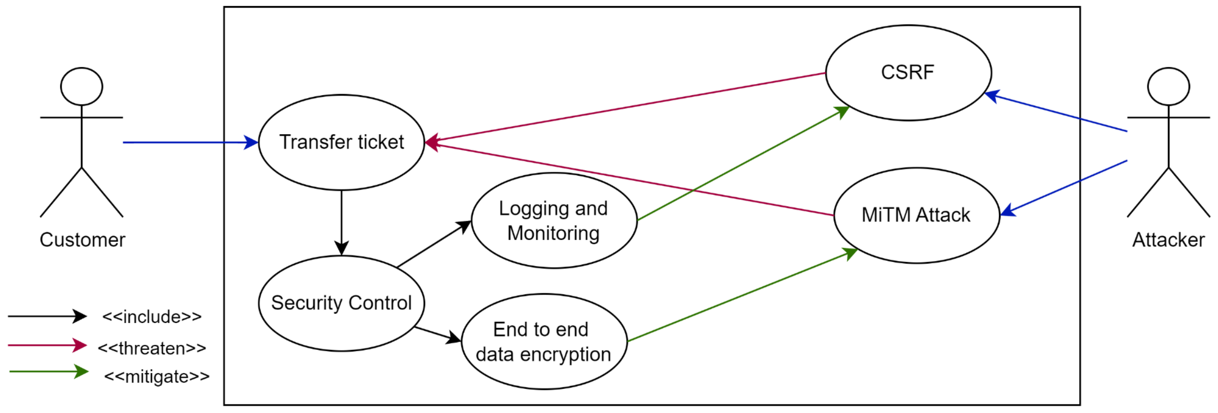

Figure 8. MC7: Transfer Ticket

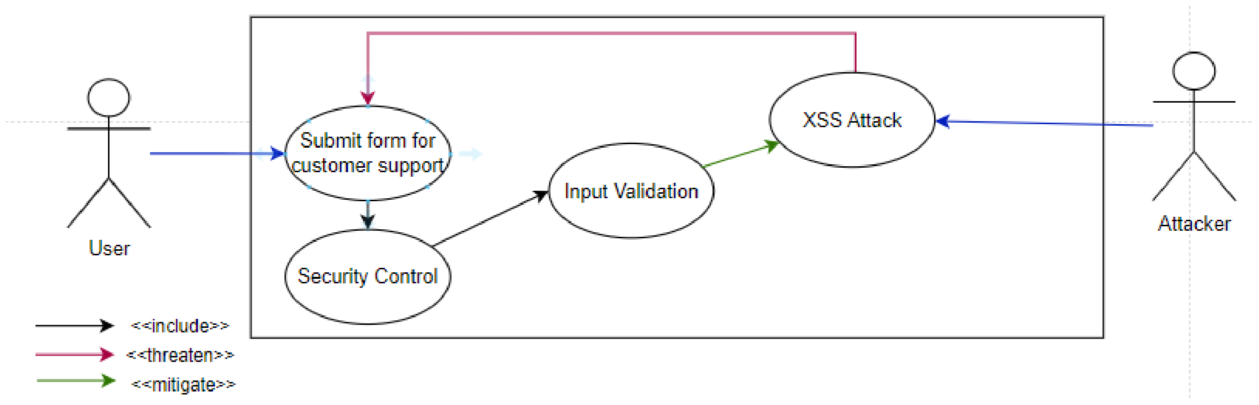

Figure 9. MC8: Customer Support

2.2 Misuse Case Description

UC	MC	MC Description
UC1 (Sign up)	MC1	Attackers attempting to exploit vulnerabilities in the sign-up process to gain unauthorized access, execute SQLi attacks or create duplicate accounts.
UC2 (Login)	MC2	Attackers attempt to exploit the login process by brute forcing the login credentials of a registered user, capturing session information, or misusing password recovery features to gain unauthorized access.
UC14 (Create events) UC15 (Edit events) UC17 (Delete events)	MC3	Attackers exploiting administrator functionalities to gain unauthorized access, escalate privileges, hijack sessions, or steal data.

UC5 (Search events)	MC4	Attackers exploiting the event search functionality to perform session hijacking, SQLi attack, or cross-site scripting.
UC7 (Raffle tickets)	MC5	Attackers exploiting the raffle ticket system to manipulate the database by performing denial-of-service (DoS) attacks, submitting duplicate entries, or manipulating/ predicting the random number generator.
UC11(Checkout)	MC6	Attackers can exploit the payment system during the checkout process by ticket fraud.
UC9 (Transfer tickets)	MC7	Attacks can exploit the system attempting to steal ticket information through ticket fraud, session hijacking, and MiTM attacks.
UC13 (Customer support)	MC8	Attackers can exploit the system by XSS attack through the customer support field.

Table 17. Misuse Case Description

2.3 Potential Risk of Application

This section goes over some of the potential risks that the application might face which cannot be prevented by developers. These include various external and internal factors that could affect the application's performance and availability. In order to be better prepared to deal with these risks and manage their impacts, the team has identified these risks in the table below, including details on each identified risk, its type, severity, and a brief description.

Risk ID	Type of Risk	Severity	Description
R1	Data Leakage	High	Data leakage can occur due to loopholes in data protection measures, which can lead to unauthorized access and exposure to sensitive information.
R2	Network Issues	High	Network failures can cause disruptions in communication between the client and server, leading to an inability to access the TicketingHuat platform.
R3	Hardware Failure	High	Hardware failures such as disk crashes or server malfunction can lead to data loss and service downtime which can impact the platform availability.

R4	Server Down	Medium	Server downtime such as overloads can prevent users from accessing the platform, leading to loss of service.
R5	Security Issues	Medium	Cyber attacks such as DDoS and unpatched vulnerabilities can compromise server integrity and availability.
R6	Environmental Factor	Low	Environmental factors including fire, flooding, earthquake, etc. can physically damage the server.

Table 18. Potential Risk of Application

2.4 Threat Modelling

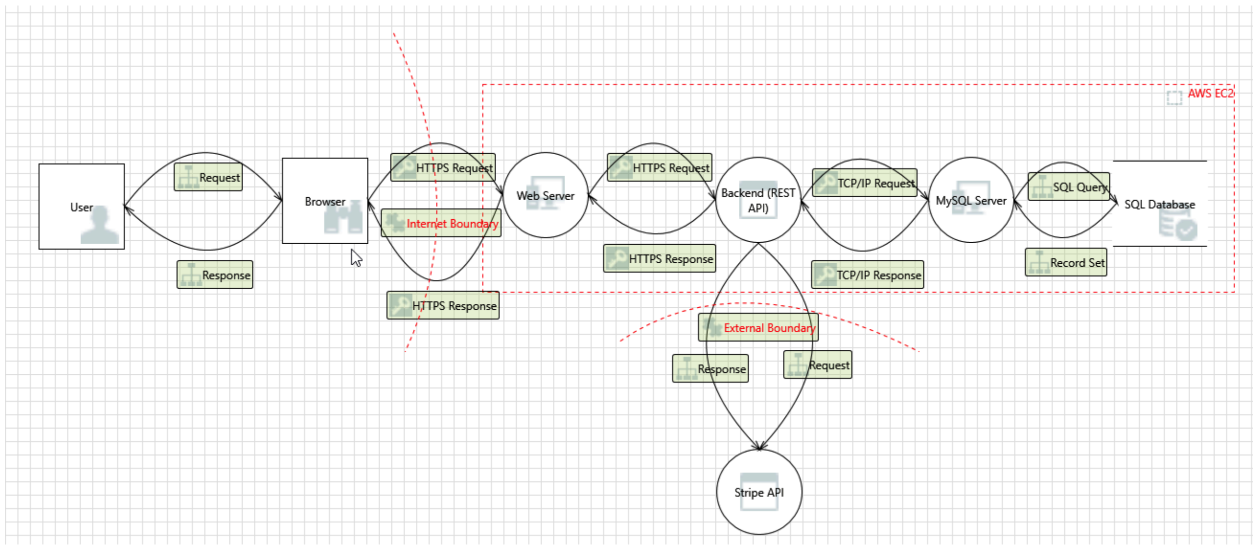

Figure 10. Level 0 Data flow Diagram

Threat modelling of TicketingHuat based on the Microsoft STRIDE model.

TH	TH Description	Actions to be taken	Category
TH1	Attackers may spoof browser external entity and this may lead to unauthorized access to the Web Server.	Implement authentication mechanisms to identify the user and external entities.	Spoofing
TH2	SQL Database may be spoofed by an attacker and this may lead to data being written to the	Restrict access to the SQL database to trusted IP addresses or subnets only, making it hard	Spoofing

	attacker's target instead of SQL Database.	for attackers to spoof the database.	
TH3	The web server could be subjected to XSS attack because it does not sanitize untrusted input.	By Implementing input validation and sanitization to ensure no malicious input is submitted to the server. Use CSP headers to prevent the execution of malicious scripts.	Tampering
TH4	The web server could be subjected to SQLi.	Use parameterized SQL queries and implement input validation and sanitization. Error messages displayed to clients should not be detailed and should not expose underlying system architecture.	Tampering
TH5	The browser and web server claims that it did not receive data from a process on the other side of the trust boundary.	Implement TLS to prevent interception and tampering of data. Implement detailed logs of all transactions to verify whether data is actually sent.	Repudiation
TH6	Improper data protection of SQL Database can allow an attacker to read information not intended for disclosure.	Conducting regular audits of database access controls and encrypting data during transit to protect sensitive data from unauthorized access	Information Disclosure
TH7	An external agent interrupts HTTPS data flowing across the trust boundaries in either direction.	Implement TLS to encrypt data across trust boundaries.	Information Disclosure
TH8	Potential excessive resource consumption for SQL server or database.	Rotate logs and data every 3 months. Index frequently used columns to speed up data retrieval.	Denial of Service
TH9	The web server might crash.	Prepare a backup of the web server so it will be easier to bring the server back up.	Denial of Service
TH10	Stripe API may be able to impersonate the context of	Implement JWT to improve authentication and authorization of REST API.	Elevation of Privilege

	REST API in order to gain additional privilege.		
TH11	An attacker exploiting trust between authenticated user browser and the web server, causing the browser to perform unauthorized actions through CSRF attack.	Generate a unique CSRF token for each user session, and validate the token on the server side for each request.	Elevation of Privilege
TH12	An attacker may conduct code injection attacks to change the flow of program execution within Web Server to the attacker's choosing.	Implementing input validation and sanitization to ensure only properly formatted data is processed. Use parameterized SQL queries or prepared statements.	Elevation of Privilege
TH13	Browsers may be able to remotely execute code for Web Server.	Implement CSP to prevent the execution of scripts. Use docker to isolate the web server, reducing the impact of potential RCE. Avoid functions that execute code dynamically. Use parameterized queries and prepared statements to prevent SQLi which can lead to RCE.	Elevation of Privilege

Table 19. Threat Modelling

2.4.1 Attack Surface Analysis

This section details the analysis of the software's attack surface, listing entry and exit points where attacks may occur. This analysis helps in identifying and securing potential points of vulnerability.

AS	Attack Surface	Entry Point	Exit Point	Threats
AS1	Sign-up	UI form submission (Sign-up form).	Success/failure on the client browser.	SQLi, MiTM, Data sniffing, DoS.
AS2	Login/Authentication	UI forms submission (Login form).	Success/failure on the client browser.	Brute Force attack, Session Hijacking, Credential Stuffing, MiTM, SQLi.

AS3	Password Reset	UI forms submission (Password Reset Form).	Success/failure message on client browser.	Phishing attacks and account takeover. MiTM, XSS, SQLi
AS4	Search Events	Search field submission.	Data retrieval and display of events.	SQLi, XSS.
AS5	Raffle Ticket	UI form submission (Email form).	Success/failure message on client browser.	DoS, Data Leakage, Submit duplicate ticket entries, database ticket manipulation, RNG manipulation/prediction
AS6	Event Management	UI Forms submission (CRUD Events).	Success or failure message for any edits on client browser.	Session Hijacking, Unauthorized access, Privilege escalation
AS7	Account Management	UI Forms submission (Read, Update, Delete Accounts).	Success or failure message for any edits on client browser.	Session Hijacking, Unauthorized access, Privilege escalation
AS8	Payment	UI Forms submission (Payment details).	Redirection to STRIPE gateway.	Guess card pin, Brute Force Attack, XSS, SQLi Attack
AS9	Customer Support	UI Forms submission (Email and description).	Success/Failure messages on success/failure of submitting inquiries on client browser.	XSS.
AS10	User Session	Successful login.	Valid user session.	CSRF, Session Fixation, Session Hijacking.
AS11	POST/GET Request	Web Server.	Data retrieval, display on client browser.	Data sniffing, Data tempering, SQL Injection, Data leakage.
AS12	Backend SQL database	SQL query to the database.	Data retrieval from the database.	Unauthorized access, data

				leakage, data tampering.
AS13	Firewall Server	Incoming/outgoing traffic.	AWS.	Unauthorized access, Data sniffing, Denial of Service.
AS14	Logging Server	Log data collection.	Log Storage, Log retrieval.	Data Tampering, Unauthorized Access, Data Leakage, DoS.

Table 20. Attack Surface Analysis

2.4.2 Security Architecture (Physical and Logical)

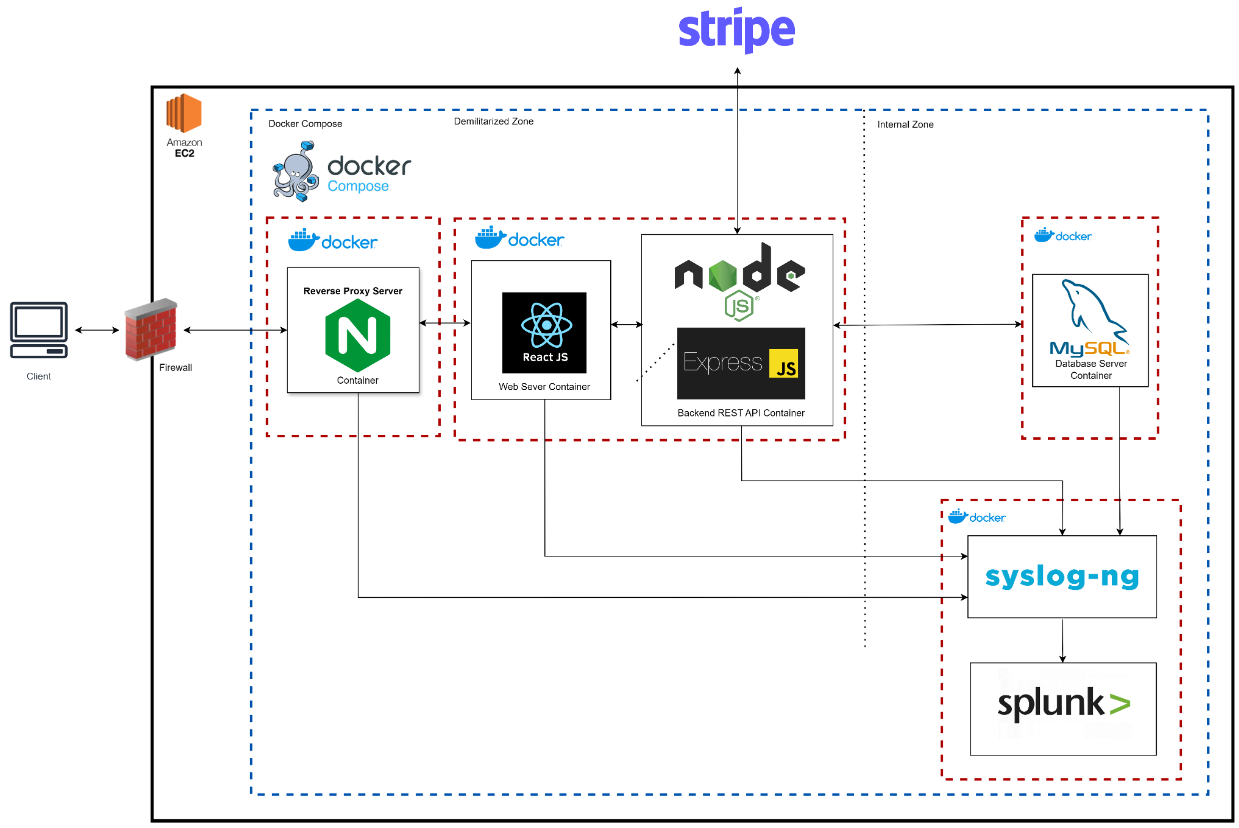

Figure 11. Physical Architecture

This diagram shows a multi-container Docker-based application deployed using the Amazon EC2 instance. It is separated into a Demilitarized Zone (DMZ) and an Internal Zone, and integrated with Stripe for payment processing. The client first communicates through the firewall to a reverse proxy (Nginx), which forwards requests to the web server (ReactJS). The front end is handled by the web server and then communicates using the backend REST API (Node.js with Express) for backend logic and API endpoints. The backend API then interacts with the MySQL database server in the Internal Zone for data storage and management. Both the web server and the backend API send log data to a syslog-ng container for centralized logging. Splunk is then used for monitoring logs through the data in syslog-ng. For payment transactions, the backend API communicates with the external Stripe service. This physical architecture ensures a secure and scalable architecture with Docker providing containerization, maintaining clear separation between external-facing and internal services.

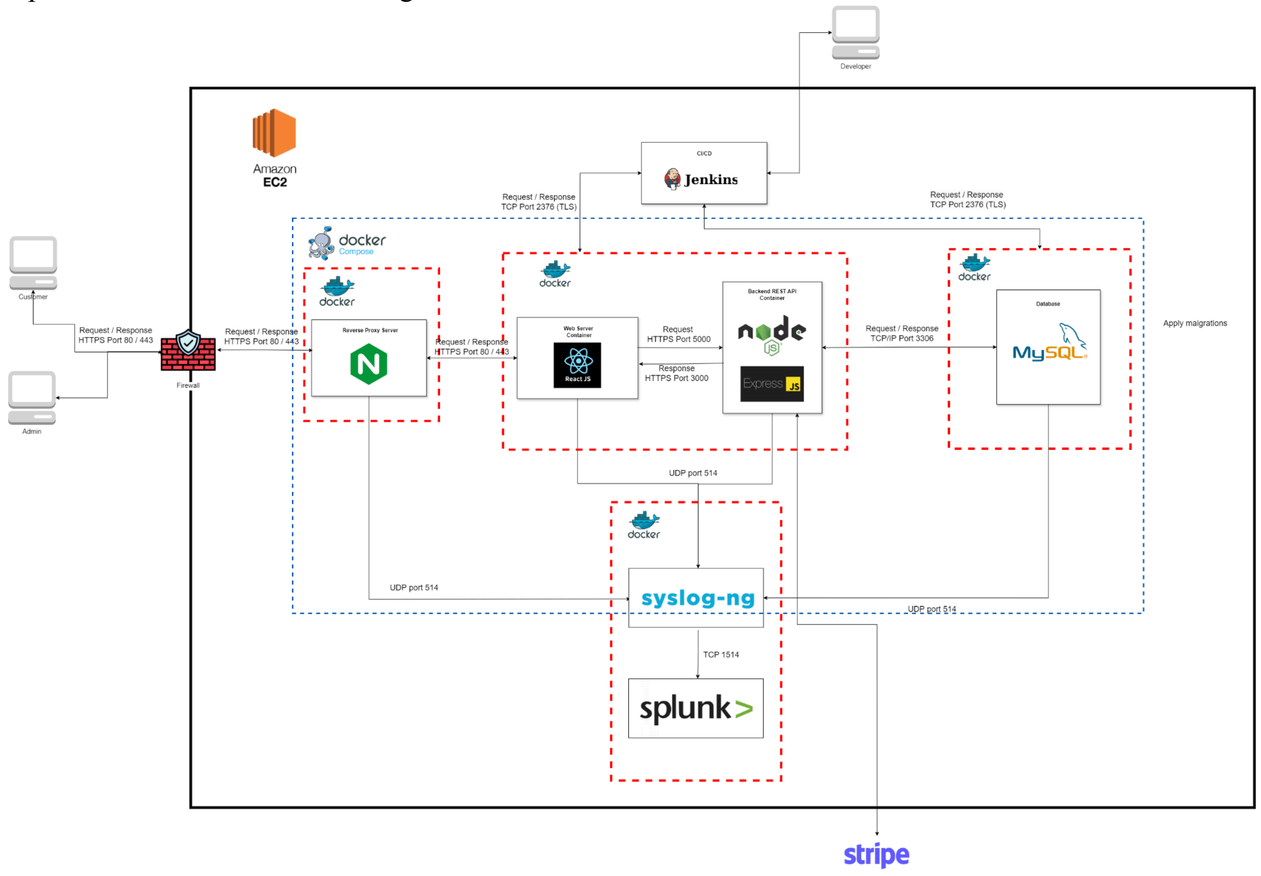

Figure 12. Logical Architecture

The diagram depicts how the web application would be structured in Amazon EC2. All components of the web application will be stored inside their own individual docker which will decrease the attack surface of the web application overall. Users will only be able to access the application through port 443 (HTTPS). If the user tries to access the application through port 80 (HTTP), they will be redirected to port 443. Nginx will be used to route the users to the ReactJS frontend, which will then communicate the backend utilising ExpressJS and NodeJS on port 5000 for business logic with a local MySQL server on port 3306 for data storage. Syslog-ng will be used to collect logs via UDP port 514 and then forward them to Splunk on TCP port 1514 for centralised log management and analysis. Jenkins will be used to facilitate all CI/CD processes over TLS on port 2376. It manages the application lifecycle and automates builds and deployment. Stripe is used for payment processing, working together with the back end. This setup ensures the web application is secure, scalable and efficient with modular and isolated services.

2.5 Security Design

Description and ID number of countermeasures placed for the use cases. Threat models and Attack Surface analysis.

2.5.1 Confidentiality Design

This section contains the security designs aimed at protecting the confidentiality of user data. This ensures that sensitive information is protected from unauthorized access.

SD	Features	Attack surface	Misuse case	Description
SD1	Data Masking	AS1, AS2, AS3, AS8.	MC1, MC2, MC6	Data masking should be implemented in the user interface to ensure that sensitive information is protected from unauthorized viewing by displaying only partial data, as recommended by NIST guidelines (*NIST Special Publication 800-63B*, n.d.).
SD2	Encryption in Transit, at rest	AS1, AS2, AS3, AS5, AS6, AS7, AS8, AS11, AS12	MC1, MC2, MC3, MC4, MC5, MC6, MC7, MC8	Data at rest will be encrypted using AES-256. This protects data stored on disks and databases from unauthorised access. HTTPS should be used to encrypt data transmitted between the user's browser and the webserver to protect sensitive information from being intercepted during transmission by utilising protocols such as TLS 1.2 or higher as recommended by NIST (*NIST Special Publication 800-63B*, n.d.). Data during processing will be encrypted through using docker to protect it from other processes.
SD3	Password with hashing salt	AS1, AS2, AS3, AS12	MC1, MC2	The user interface should implement password hashing with a 32-bit salt using PBKDF2 with an iteration account of 100,000 to ensure secure password storage and protect against brute force attacks which are recommended by NIST guidelines (*NIST Special Publication 800-63B*, n.d.).
SD4	Password Policy	AS1, AS2, AS3, AS12	MC1, MC2	The system shall only allow passwords at a minimum of 8 characters, max of 12 characters, with a mix of uppercase letters,

				lowercase letters, numbers, and special characters. This is to protect against brute-force attacks, which is recommended by NIST (*NIST Special Publication 800-63B*, n.d.). Users will also be prompted to change their password every 3 months.
SD5	Storage of sensitive data	AS1, AS2, AS6, AS7, AS12, AS14	MC1, MC2, MC3	Encrypt sensitive data stored in databases and backup storage. Use database encryption features and ensure backup files are encrypted using AES-256.

Table 21. Confidentiality Security Design

2.5.2 Integrity Design

This section contains the security designs aimed to ensure data integrity so as to prevent data tampering and ensure the accuracy and consistency of the platform's data.

SD	Features	Attack surface	Misuse case	Description
SD6	Input Validation	AS1, AS2, AS3, AS4, AS5, AS8, AS9	MC1, MC4, MC6, MC7, MC8	Input validation, such as SQL parameterized query should be implemented on client and server side to ensure that all incoming data is validated before it is processed by the application.
SD7	Input Sanitization	AS6, AS7, AS8, AS9	MC4, MC6, MC8	Input Sanitization should be implemented to remove harmful input data that could be exploited to perform malicious actions such as XSS.
SD8	Resource Locking	AS2, AS5, AS6, AS7, AS8, AS10, AS11, AS12	MC2,MC3, MC5, MC6, MC7	Resource locking should be implemented by making sure that any authenticated user cannot have multiple sessions.

Table 22. Integrity Security Design

2.5.3 Availability Design

This section contains the security designs aimed at ensuring the platform's high availability and resilience so as to maintain continuous operation.

SD	Features	Attack surface	Misuse case	Description
SD9	Rate limiting	AS2, AS5, AS 8,	MC2, MC6, MC7	Rate limiting should be implemented in the backend to ensure that DoS/DDoS/brute force attacks are blocked within the first 10 connections and brute force is mitigated by locking an account for 5 minutes after 5 failed login attempts and 5 failed PIN number during payment.
SD10	Synchronous Data Replication	AS1, AS2, AS3, AS12	MC1, MC2, MC5, MC6, MC7	Implement synchronous data replication to ensure data consistency and high availability. This involves writing data to multiple locations to prevent data loss and ensure all replicas are up-to-date.
SD11	Failover	AS1, AS2, AS3, AS5, AS8, AS11 , AS12	MC1, MC2, MC5, MC6, MC7	Implement failover to ensure continuous operation. This involves having a backup server, software or network that takes over automatically if the main system fails to ensure everything keeps running smoothly without interruption.

Table 23. Availability Security Design

2.5.4 Authentication Design

This section contains security designs aimed at ensuring secure measures for user authentication.

SD	Features	Attack surface	Misuse case	Description
SD12	MFA	AS2, AS5, AS6, AS7, AS8	MC2, MC3	Implement MFA to gain access to the system to reduce risk of attackers login by guessing, or using stolen or leaked data.
SD13	Sessions	AS2, AS5, AS6, AS7, AS8, AS10,	MC2, MC3, MC5, MC6, MC7	Use secure HttpOnly cookies to store JWTs, and invalidate sessions upon logout and timeout. Timeout after 30 minutes, which will require users to re-authenticate.

Table 24. Authentication Security Design

2.5.5 Authorization Design

RBAC will be implemented to ensure that users only have access to different types of data based on their user roles. RBAC prevents unauthorized access and reduces the risk of data breaches and other security incidents (*Role Based Access Control*, 2024).

SD	Features	Attack surface	Misuse case	Description
SD14	RBAC	AS2, AS5, AS6, AS7, AS8	MC2, MC3, MC5, MC6, MC7	RBAC is implemented so that each staff user has their set of responsibilities. In the event that the account is compromised, it limits the damage.

Table 25. Authorization Security Design

The roles defined in the system are Admin, Event Staff, Customer Support Staff, Database Admin, Customer and Guest. Each user role contains different permissions to which they can access the data. The details can be found in the following Data Access Control Matrix.

User Roles	Role Description
Admin	Authenticated user with permission to mainly manage customer data and event data
Event Staff	Authenticated user with permission to mainly manage event data
Customer Support Staff	Authenticated user with permission to mainly read customer data in order to support Customer’s permission
Database Admin	Authenticated user with permission to mainly manage database schema
Software Engineer	Authenticated user with permission to mainly manage codebase data
Customer	Authenticated user with permission to mainly create an account and purchase tickets
Guest	Unauthenticated user with permission to only search and view events

Table 26. User Roles and Descriptions

Data	User Roles						
	Admin	Event Staff	Customer Support Staff	Database Admin	Software Engineer	Customer	Guest
Customer Data	C, R, U, D	R	R	R	R	C, R, U	-
Event Data	C, R, U, D	C, R, U, D	R	R	R	R	R
Ticket Data	R	R	R	R	R	R	-
Order Data	R	R	R	R	R	R	-
Log Data	R	-	-	-	-	-	-
Codebase Data	-	-	-	-	C, R, U	-	-
Database Schema	-	-	-	C, R, U, D	-	-	-

Table 27. Data Access Control Matrix

2.5.6 Accountability Design

This section contains security designs to ensure accountability through secure logging and monitoring of user actions and system events.

SD	Features	Attack surface	Misuse case	Description
SD15	Logging	AS1, AS2, AS3, AS5, AS6, AS7, AS8, AS12, AS13, AS14	MC1, MC2, MC3, MC5, MC6, MC7	Implement secure and comprehensive logging and monitoring. Log activities such as (user actions, payment, administrative actions and failed login attempts) and specify the details captured such as the (email, action type, timestamp, and IP address where the requests originated from). If sensitive data is needed, it should be hashed or masked. (*Logging - OWASP Cheat Sheet Series*, n.d.) Logs should be rotated every 3 months.

Table 28. Accountability Security Design

This section provides a detailed diagram of the database schema used in the software. It outlines the structure of the database, ensuring efficient data organization and retrieval while maintaining data integrity and security.

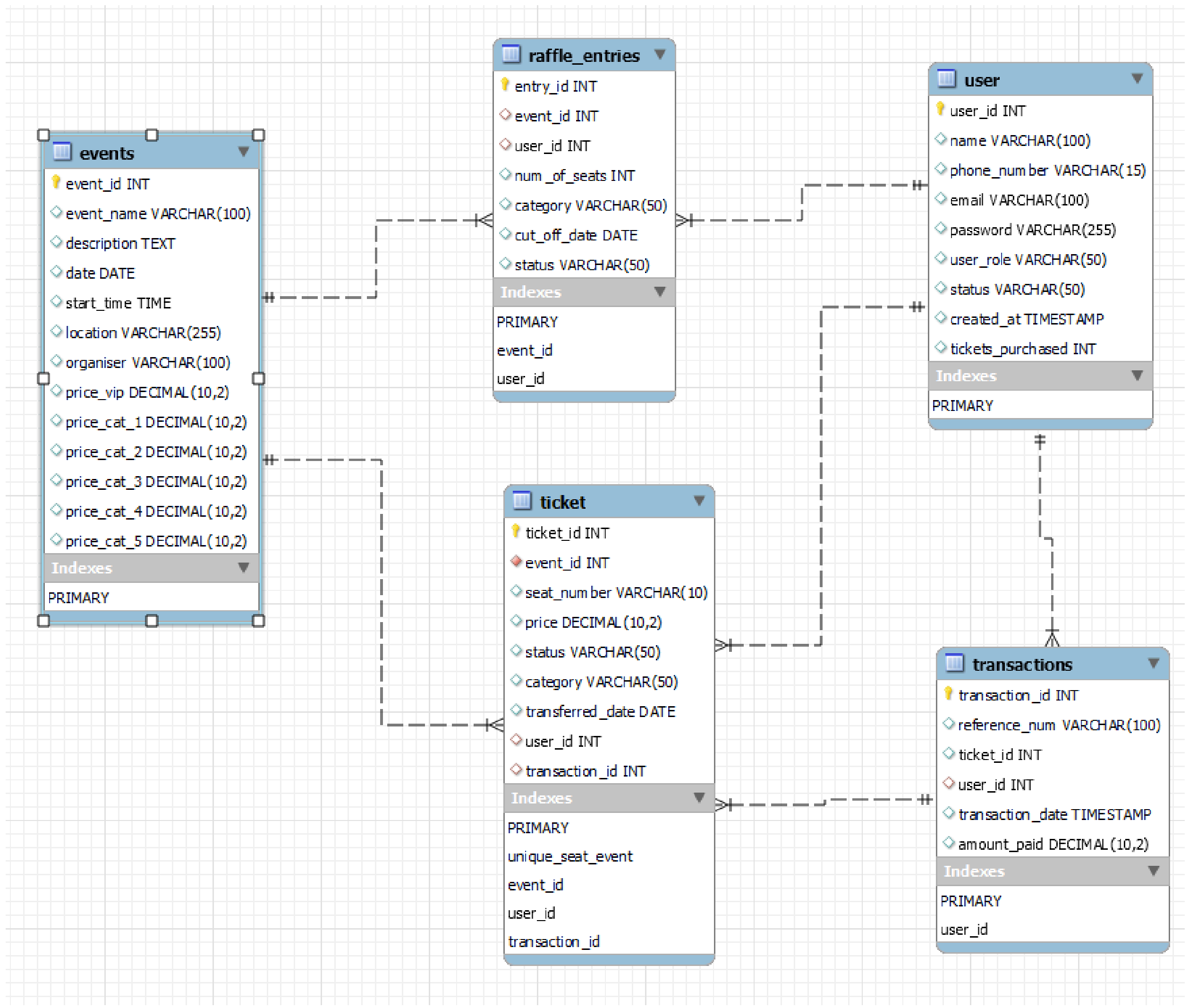

Figure 13. Database Schema

References

[1] NIST Special Publication 800-63B. (n.d.). https://pages.nist.gov/800-63-3/sp800-63b.html

[2] Logging - OWASP Cheat Sheet Series. (n.d.). https://cheatsheetseries.owasp.org/cheatsheets/Logging_Cheat_Sheet.html

[3] Role based access control. (2024, June 10). Cisco. https://www.cisco.com/c/en/us/td/docs/routers/sdwan/configuration/system-interface/ios-xe-17/systems-interfaces-book-xe-sdwan/m-rbac-17-13.html

Appendix

Words	Meaning
MiTM Attack	Man in the middle attack
SQLi Attack	SQL injection attack
DoS Attack	Denial-of-service attack
RBAC	Role-based access control
WAF	Web Application Firewall
XSS	Cross-Site
CSRF	Cross-Site Request Forgery
CDN	Content Delivery Network
MFA	Multi-Factor Authentication
DNS	Domain Name Server
RBAC	Role-Based Access Control
WFA	Web Application Firewall
CRUD	Create, Read, Update, Delete
OTP	One Time Password
CSA	Cyber Security Agency of Singapore’s
PBKDF2	Password-Based Key Derivation Function 2
TLS	Transport Layer Security

PDPA	Personal Data Protection Act
PIN	Personal Identification Number
API	Application Programming Interface
OWASP	Open Worldwide Application Security Project
SQL	Structured Query Language
HTTPS	Hypertext Transfer Protocol Secure
JWT	JSON Web Token
CSP	Content Security Policy
RCE	Remote Code Execution

Table 29. Data Dictionary