# Software Architecture of Train Inter Payment System (TrIP)

Group 04

April 5, 2024

# Contents

# 1 Product Introduction

The Train Inter Payment System (TrIP) is a collaborative project initiated by three railroad tycoons to streamline the payment process for train travel. These tycoons, operating a network connecting towns, industries, and a university, aim to address the interoperability issues of their existing payment systems. TrIP will feature smart payment terminals at each station, enabling direct communication for subscription validation or single-fare payments. This system, underpinned by a service-based architecture, involves critical components like payment terminals and tycoon-specific systems. It's designed with key stakeholder requirements in mind: maintainability and operational efficiency for the owner, usability and reliability for the tycoons, and usability and security for passengers. The project seeks to ensure passengers can easily manage payments and subscriptions across the network, enhancing the overall travel experience while safeguarding passengers data.

# 2 Decisions

## 2.1 Decision 1: User Interface

### Status

Accepted. Reviewed after the decision to include a smartphone application and a web application.

### Architectural Summary

| In the context of | Designing a user interface for the TrIP system payment terminals |
|---|---|
| **Facing** | The challenge of letting various type of passengers easily find information and make purchases, |
| **To achieve** | Usability and availability for passengers, maintenability and low operational costs for the TRiP owner |
| **We considered** | Option 1: Use of Standardized UI Components; Option 2: Custom Designed Interactive Interfaces; Option 3: Open Source Frameworks; Option 4: Proprietary High-End Frameworks |
| **And decided for** | Option 1: Use of Standardized UI Components |
| **Because** | It offers a balance of development speed, cost, consistency, and usability that meets our criteria and constraints |
| **Accepting** | Potential dependency on external communities and companies. |

### Concern

Passengers require a user interface that is easy to navigate, visually appealing, and provides a seamless experience across different tycoon systems. we face the challenge of selecting guiding principles and frameworks that will shape the user experience and interaction design. Related user stories are listed here:

- **User Story 16:** As a passenger, I want to be able to purchase a single ticket that allows me to travel across all train networks so that I can travel seamlessly without needing to buy separate tickets for each network.,

- **User Story 7:** As an advocate for accessibility, I want the system to offer voice-activated features and screen reader compatibility for passengers with visual impairments so that the service is inclusive and accessible to everyone.,

- **User Story 14:** As a visually impaired person, I want a high contrast interface and big font, so that I can avoid mistakes when buying my tickets.,

- **User Story 9:** As a tourist, I want to choose among many languages, so that I don't have to translate using my phone.,

- **User Story 29:** As the TrIP owner, I want the system to operate with the lowest possible operational costs, so that it remains cost-effective over time.,

- **User Story 30:** As the TrIP owner, I want the system maintenance efforts to be easily scalable, so that costs do not increase disproportionately as we expand.,

### Context

We need to design an intuitive and engaging user interface for the TrIP system terminals, web application and smartphone application. The design of the user interface on the terminals involves the passenger's interaction with the system, from querying routes to finalizing ticket purchases.

### Criteria

The decision will be guided by the following criteria:

- *Usability*: Consistency in design to provide a unified look and feel across all terminals, web apps and smartphone apps.

- *Usability* and *Availability*: Accessibility to ensure the system is usable by all passengers, including those with disabilities, at all time.

- *Maintenability*: Ease of maintenance and scalability for future enhancements.

- *Cost effectiveness*: Low operational cost.

- *Flexibility*: Responsiveness so that the interface can adapt to various screen sizes and orientations.

## Option 1: Use of Standardized UI Components

This approach involves adopting a comprehensive design system, such as Google's Material Design or IBM's Carbon Design System, which offers a robust set of standardized UI components. These components include buttons, forms, toggles, navigation patterns, and more, all designed with consistency and usability in mind. By utilizing these pre-designed components, the development process can be significantly accelerated, as developers and designers will not need to create common UI elements from scratch. This ensures a cohesive look and feel across the entire application, enhancing the user's ability to intuitively navigate the system.

- **Pro:** Significantly reduces development time and ensures UI consistency.

- **Pro:** Both Google's Material Design and IBM's Carbon Design System are open source, hence they would allow developers to easily debug the interface, identify possible bugs, seek for contributions online or even contribute themselves.

- **Con:** May limit unique branding opportunities and design customization.

- **Con:** Some developers may consider it a limit for their creativity.

## Option 2: Custom Designed Interactive Interfaces

This option focuses on creating bespoke interactive interfaces from the ground up, specifically tailored to the unique needs and brand identity of the TrIP system. This could involve developing custom animations, unique layout designs, and interactive elements that engage users in a novel way. By focusing on custom designs, the TrIP system can distinguish itself from competitors and provide a unique user experience that directly addresses specific user needs and preferences.

- **Pro:** Allows for full creative freedom and the opportunity to innovate.

- **Pro:** Less operational cost, and higher maintainability.

- **Con:** More time-consuming and expensive due to the bespoke nature of the design and development process.

- **Con:** Expertise within the developing team is needed, possibly more developers. This might offset the lower operational cost.

## Option 3: Open Source Frameworks

Utilizing open-source UI frameworks such as Bootstrap, Foundation, or Vue.js offers a middle ground between complete customization and strict standardization. These frameworks are supported by large communities of developers, ensuring that the frameworks are well-documented, frequently updated, and robust against common web development challenges. They come with a variety of UI components that can be easily modified to fit the system's needs, providing both speed in development and a degree of customization.

- **Pro:** Combines rapid development with the flexibility of customization.

- **Con:** Might still require significant effort to stand out from the default "framework look."

- **Con:** Possible discontinuation.

## Option 4: Proprietary High-End Frameworks

Choosing proprietary frameworks such as Telerik, DevExpress, or Adobe XD's design systems offers access to a suite of advanced features, including sophisticated data visualization tools, complex UI components, and comprehensive support services. These frameworks are often optimized for performance and come with extensive documentation and professional support, ensuring that the development team can create a high-quality user interface while potentially saving time on troubleshooting and problem-solving.

- **Pro:** Provides a wide range of advanced features and dedicated support.

- **Con:** Incurs additional costs due to licensing fees and may lock the project into a specific vendor or technology stack.

- **Con:** Train system doesn't have to be that compilicated, high-end products can be redundant, also the passengers want us to keep it simple.

## Decision

Option 1 is chosen. This option lifts a lot of weight from the development team. This reduces operational cost and enhances maintenability, which are the main concerns of the TrIP owner. Furthermore, these interfaces are well tested and user-friendly, making them a natural choice to satisfy the need for usability of the passengers and the reliability requested by the tycoons. Generally speaking, user interfaces for train systems are not sophisticated enough to require more expensive and complicated set-ups. A careful user testing is strongly adviced, to choose a suitable setup.

## Consequences

**Positive Consequences:**

- Access to a broad community for support and troubleshooting.

- Cost savings by avoiding licensing fees associated with proprietary software.

- Rich ecosystem of plugins and extensions to enhance functionality.

- Frequent updates and a large pool of developers familiar with the frameworks.

**Negative Consequences:**

- Potential dependency on external communities for critical updates and support. Some of these might require expensive professional support.

- Risk of choosing a framework that may not align with long-term technology trends or become obsolete. Information sourcing about the future of the chosen project is crucial.

- Need for rigorous selection to ensure accessibility and responsiveness standards are met. This would be a consequence of any of the choices listed.

## 2.2   Decision 2: Database technology

**Status**

Accepted.

### Architectural Summary

| | |
|---|---|
| **In the context of** | choosing appropriate database technology to store information about travels and payments, |
| **Facing** | the challenge of storing data coming from different tycoons systems in a cost-effective manner, |
| **To achieve** | *Mainteinability*, *security*, *cost efficiency*, *performance* and *scalability* |
| **We considered** | Option 1: SQL Database (e.g., PostgreSQL); Option 2: NoSQL Database (e.g., MongoDB); |
| **And decided for** | Option 1: SQL Database (e.g., PostgreSQL) |
| **Because** | It suits the kind of data we need to store, it is easier to source developers expertise and security is well tested, |
| **Accepting** | Potential difficulties in horizontal scaling, less flexibility in the data model. |

### Concern

The main staleholder's concerns related to this decision are the passengers request to have a good integration among different tycoons subscription, the TrIP owner request to keep maintenance costs low and the data protection required by government and passengers.

Related user stories are listed here:

- **User Story 18:** As a government regulator, I want the payment system to adhere to data protection and financial transaction security standards so that passengers' personal and financial information is secure.,

- **User Story 29:** As the TrIP owner, I want the system to operate with the lowest possible operational costs, so that it remains cost-effective over time.,

- **User Story 30:** As the TrIP owner, I want the system maintenance efforts to be easily scalable, so that costs do not increase disproportionately as we expand.,

- **User Story 32:** As the TrIP owner, I want a cost-effective database solution for data management, so that storage and access costs remain low.,

### Context

In developing the TrIP system we are faced with the decision of choosing an appropriate database technology. This choice hinges on our need to ensure data integrity, support complex queries for transaction processing, and maintain scalability and security. The database must handle a wide array of data, including user subscriptions, fare transactions, and station and route information, necessitating a robust system that supports complex queries and relational data structuring. The architecture might include more than one databases, depending on the needs that will arise during later decisions.

### Criteria

- *Cost efficiency* and *mainteinability*.

- Comprehensive *security* features to safeguard sensitive data.

- Data integrity and transactional consistency for financial transactions.

- Ability to support complex queries and relational data models.

- *Scalability* to grow with the system's user base and data volume.

- *Performance* under varying load conditions.

### Option 1: SQL Database (e.g., PostgreSQL)

A relational database model renowned for its strong consistency, ACID (Atomicity, Consistency, Isolation, and Durability) compliance, and the ability to efficiently handle complex queries and data relationships.

- **Pro:** High data integrity and robust support for complex relational data structures.

- **Pro:** We have a highly structured data.

- **Pro:** More developers are familiar with it, more resources on the topic. It has a strong community and over 30 years of active development.

- **Pro:** Good with concurrency.

- **Pro:** Well tested security features like encryption, access control, SQL injection prevention, auditing capabilities.

- **Pro:** Open source and free.

- **Con:** Scalability challenges in horizontally distributed architectures compared to NoSQL options.

- **Con:** Requires accurate upfront planning of the data model due to its structured nature, thereby limiting flexibility.

### Option 2: NoSQL Database (e.g., MongoDB)

A distributed database system designed for scalability and flexibility, suitable for handling large volumes of diverse data types.

- **Pro:** Offers superior scalability and flexibility for managing unstructured or semi-structured data.

- **Pro:** Enhances performance for non-structured data.

- **Con:** May compromise transactional integrity and consistency in favor of performance and scalability.

- **Con:** Relatively new, hence less well tested for security concerns.

### Decision

After thorough consideration, the decision is to implement an **SQL database**, specifically PostgreSQL for it being open source, for the TrIP system. This decision is underpinned by the SQL database's unmatched data integrity, support for complex transactions, and relational data modeling capabilities, which are crucial for the financial transactions and data relationships inherent in the TrIP system. Furthermore, train payment data are by nature very structured, don't give much creativity to the passengers, thus a relational database seems a more natural choice. Given the higher familiarity of developers, this choice is good for the QAs favoured by the TrIP owner, such as:

- Maintainability (priority 2). The owner wants a minimum amount of effort to maintain and build the system

- Operational costs (priority 2). The operational costs of the system should be as low as possible.

### Consequences

**Positive Consequences:**

- Ensures high levels of data integrity and transactional consistency, critical for financial data and user subscriptions.

- Facilitates complex data queries and relationships, enabling sophisticated data analysis and reporting.

- Provides robust security features to protect sensitive data and comply with data protection regulations.

**Negative Consequences:**

- May require additional strategies for scaling horizontally, such as implementing read replicas or sharding, to manage large data volumes and high traffic loads effectively.

- Could necessitate more intensive resource management and optimization to ensure performance at scale.

- A less flexible data model.

Choosing an SQL database aligns with the TrIP system's core requirements for data integrity, relational data handling, and transactional consistency. This foundation will support the system's initial functionality and long-term growth, with a focus on maintaining data accuracy and trustworthiness.

## 2.3 Decision 3: Routes management

**Status**

Accepted.

### Architectural Summary

| In the context of | Allowing terminals to present route options to passengers when they select their departure and arrival stations, |
|---|---|
| Facing | The challenge of letting passengers choose between different travel combination involving multiple tycoons, |
| To achieve | Integration with multiple tycoon systems, and accurate representation of options based on multiple factors, |
| We considered | Option 1: Direct Tycoon Integration; Option 2: Centralized Route Management Module; |
| And decided for | Option 2: Centralized Route Management Module |
| Because | It simplifies data flow and integration between the TrIP system and tycoon's systems, it improves scalability and maintainability of the system, |
| Accepting | Initial development effort and potential difficulties in data synchronization across modules. |

### Concern

The concern is to facilitate a seamless travel planning experience for passengers by ensuring the system can accurately and efficiently gather available route options and present them based on the user's criteria of price, time, and subscription status. Related user stories are listed here:

- **User Story 16:** As a passenger, I want to be able to purchase a single ticket that allows me to travel across all train networks so that I can travel seamlessly without needing to buy separate tickets for each network.,

- **User Story 26:** As a maintenance team member, I want the system to allow for temporary fare adjustments or bypass options during maintenance work so that we can minimize inconvenience to passengers.,

### Context

The TrIP system needs to be able to compute optimized travel options (for instance best prices and lowest travel times), gathering information from multiple tycoon systems in order to align with passenger preferences and existing subscriptions. The decision is centered on the system's interface with tycoon systems to retrieve and optimize route data, which impacts the functionality and performance of the terminal's route planning features for the passengers.

### Criteria

The key criteria for the decision include:

- *Functionality and usability*: Comprehensive and diverse route options.

- *Functionality*: Accurate representation of options based on multiple factors.

- *Integrability*: Streamlined integration with multiple tycoon systems.

### Option 1: Direct Tycoon Integration

Terminals directly interface with each tycoon system and the central database to collate route options. The route optimizer processes this data to present optimal travel solutions. This requires our system to have APIs to query each of the tycoons systems.
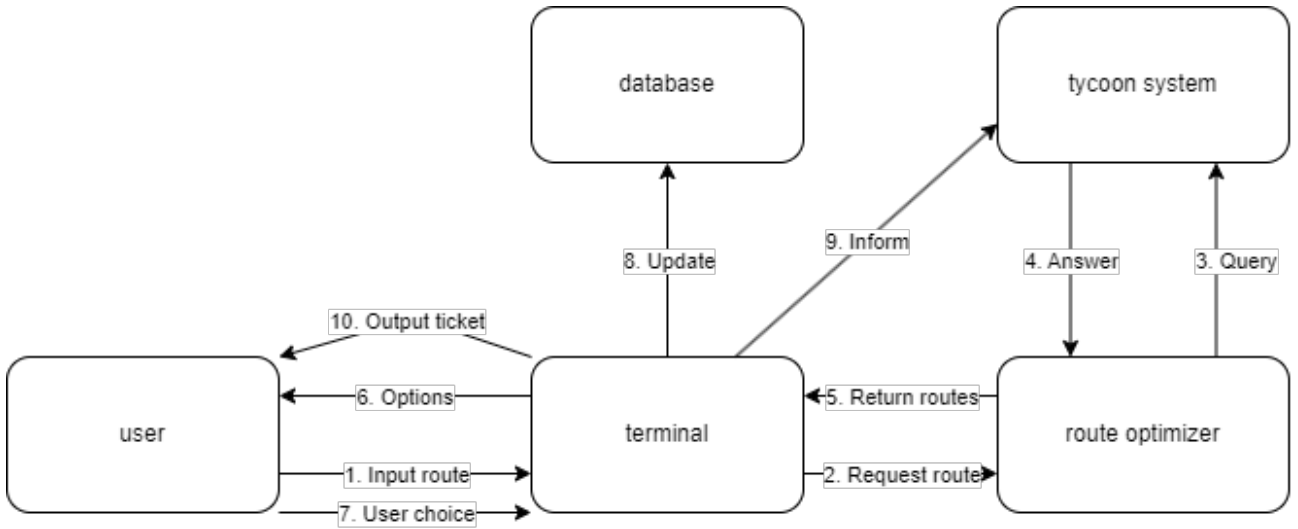
Figure 1: Direct Data Management Interface

## Option 2: Centralized Route Management Module

A central route data management module acts as an intermediary between terminals and tycoon systems, standardizing and aggregating data before it is processed by the route optimizer. A database containing the timetables will be kept up to date by the tycoon (possibly though an API, this will be the focus of a later decision). The route data management system might cache optimized routes, in order to minimize database requests. A module specialized in running optimizations with the data it is provided with interacts solely with the Route Management Module.

## Decision

We have decided to proceed with Option 2: Centralized Route Management Module. This decision is based on the module's ability to simplify the data flow between systems and to effectively manage the complexity of integrating with multiple tycoon systems. The introduction of a separate data management module will allow for greater flexibility and scalability.

## Consequences

**Positive Consequences:**

- Simplified data flow between the trip system and tycoons.

- Improved scalability and maintainability of the system.

- Easier to integrate with current and future tycoon systems.

**Negative Consequences:**

- Initial development and integration effort for the new module.

- Potential complexity in data synchronization between modules.

This approach is expected to provide a solid foundation for the system's scalability and adaptability to evolving requirements and stakeholder needs.
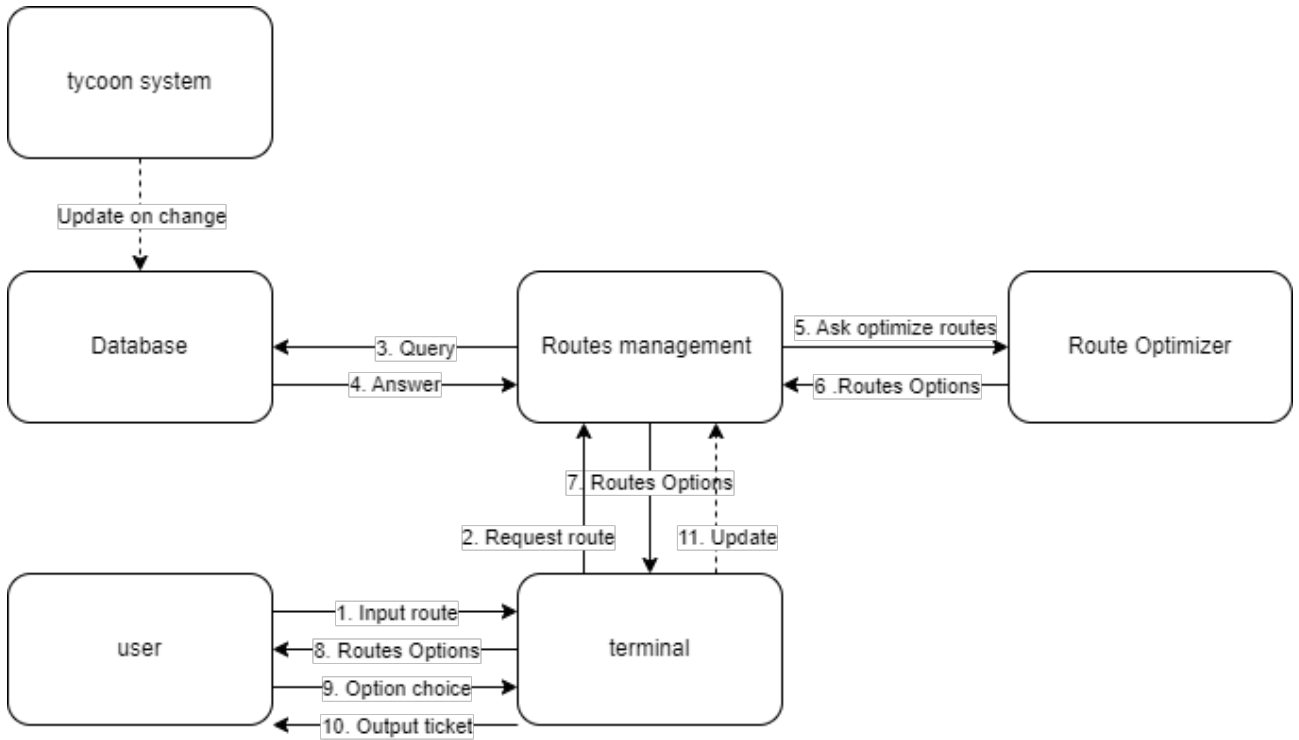
Figure 2: Centralized Route Management Module. Dashed lines represents steps to be explained in future decisions.

## 2.4 Decision 4: Account and Subscriptions management

### Status

Accepted.

### Architectural Summary

| In the context of | managing subscriptions and accounts of passengers, |
|---|---|
| Facing | the challenge of integrating many different subscriptions for the same passenger and letting subscription separated, |
| To achieve | streamlined integration with multiple tycoon systems, high passenger usability, ease of maintenance, |
| We considered | Option 1: Introduction of a subscription manager tool.; Option 2: Integrated Subscription-Route Optimization Service; Option 3: Include a Single-Ticket management module |
| And decided for | Option 1: Introduction of a subscription manager tool; and Option 3: Include a Single-Ticket management module |
| Because | It enhances scalability and maintainability though modularity, it reduces system-wide failures risks, allows occasional travellers to easily use the system, |
| Accepting | Increased System Complexity, higher Initial Costs, Data Consistency Challenges. |

### Concern

A passenger wants to traver around the network using its subscription(s) instead of always having to buy a single fare. Tycoons have explicitely requested to maintain their own subscriptions separated. The following user stories are tied to this decision:

- **User Story 16:** As a passenger, I want to be able to purchase a single ticket that allows me to travel across all train networks so that I can travel seamlessly without needing to buy separate tickets for each network.,

- **User Story 23:** As a tycoon, I want the payment system to integrate with my existing infrastructure with minimal disruption so that I can maintain high service levels during the transition.,

- **User Story 30:** As the TrIP owner, I want the system maintenance efforts to be easily scalable, so that costs do not increase disproportionately as we expand.,

Specific concerns identified for the system include the following:

- A passenger should be able to have different subscriptions for the three tycoons.

- Each of the three tycoon wants to have its own subscription fee system.

## Context

In developing the TrIP system, we explore subscription model architectures that align railway tycoons' need for flexibility with passengers' demand for simplicity. We assess the trade-offs between unified and tycoon-specific models to enhance both operational autonomy and passenger convenience. When passengers want to go from station A to station B, they want to be able to use all the subscription they have and pay for the trains belonging to tycoons they are not subsribed to. The terminal has to communicate with the tycoon systems to verify subscriptions and check routes. Alternatively, if passengers have active subscriptions to a tycoon, they should be allowed onboard the train without a ticket.

## Criteria

- *Usability*: Ease of use for the passenger.

- *Functionality*: Multiple route and subscriptions options for the passenger without excessive effort.

- *Functionality*: The system returns to the passenger usable options based on price/time/availability/subscription.

- *Integrability*: Simple integration with the tycoon systems.

- *Maintainability and scalability*: Ease to accomodate changes in the subscriptions options from tycoons.

## Option 1: Introduction of a subscription manager tool, route management needs to take passenger subscriptions into consideration.

We want to include a Subscription Manager module to our functional view. Such a module has the responsibility to verify subscriptions, communicating with the tycoon system. It should also allow passengers to create their own TrIP account and tie it to subscriptions. Since we need scalability, we consider using a TrIP account that is tied to one or many subscriptions, and let the passenger add new or automatically renew the existing ones. This means that we need an authentication means, like a magnetic card, or a phone app linked to NFC scanning or other alternatives. This will be the topic of a future decision. Furthermore, the Route management module needs to be able to optimize with respect to price, given that the passenger holds some subscription. The idea is to add an optional initial iteration to the sequence of actions sketched in decision 3. This requires a way to communicate the subscription to the terminal, be it a QR code, a code or a magnetic card. This should be the topic of another decision.

## Pros

- **Modular Architecture:** The separation of subscription management and route optimization into distinct modules enhances the system's modularity, making it easier to maintain, update, or replace parts of the system without affecting others.
- **Scalability:** Designed with scalability in mind, allowing for easy addition and management of new subscriptions or renewal of existing ones, which can accommodate growing passenger demands and evolving business requirements.
- **Reduced Risk of System-wide Failures:** By distributing functionalities across different systems or modules, the impact of a failure in one component is limited, reducing the risk of system-wide outages and improving overall system reliability.
- **Flexibility in Passenger Authentication:** Supports various means of passenger authentication (e.g., magnetic card, smartphone app), offering flexibility and convenience for passengers to access their subscriptions and travel seamlessly.
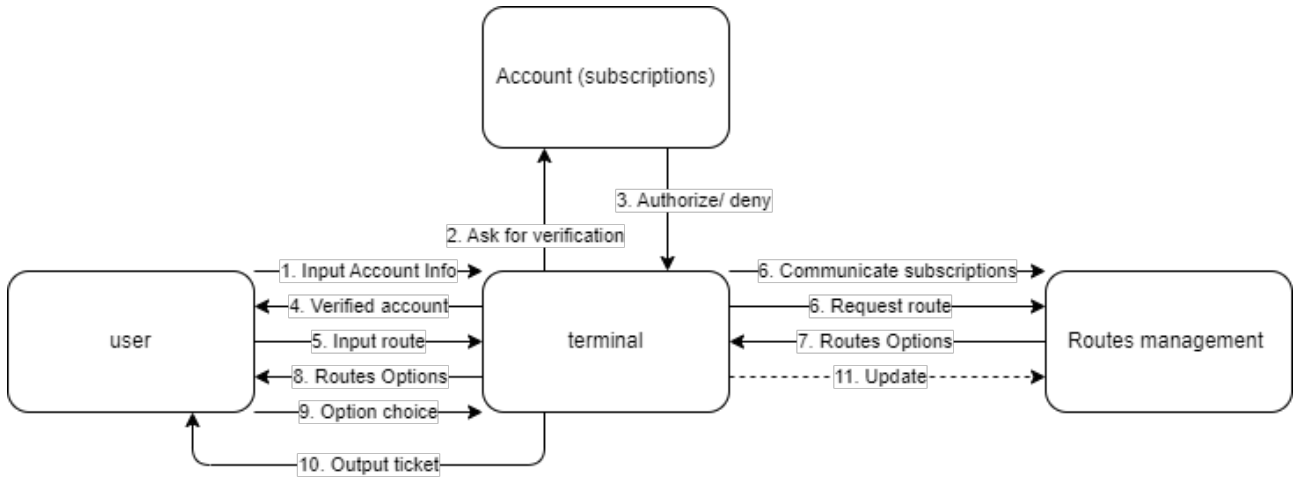
Figure 3: Account Management Interaction for single-ticket.

## Cons

- **Increased System Interactions:** The decoupled nature of the system requires more interactions between separate modules (e.g., subscription verification and route optimization), which can increase the complexity of integration and potentially lead to higher latency in response times.
- **Integration Complexity:** Ensuring seamless communication and data consistency between different modules can introduce additional complexity in system integration and require more sophisticated coordination mechanisms.
- **Higher Overhead:** Managing separate systems for subscription and route optimization may lead to higher operational overhead in terms of both system resources and administrative efforts to maintain multiple components.

## Option 2: Integrated Subscription-Route Optimization Service

The Integrated Subscription-Route Optimization Service (IS-ROS) combines the functionality of subscription management and route optimization into a single service.

### Pros

- **Performance** Streamlines the process by combining two functionalities, potentially reducing response time for route optimization.
- **Reduced Interactions between systems** Simplifies the architecture by reducing the number of interactions between separate systems.

### Cons

- **Increased Complexity within Single Module** Increases complexity within a single system, which may require more resources to develop and maintain.
- **Failures have larger impact** May lead to higher dependency on a single system, which can be a point of failure if the system goes down.

## Option 3: Include a Single-Ticket management module

Abstractly, a single ticket can be seen as an account that expires once the journey is completed. As an account, it is linked to a passenger via their personal data, if it is related to a booking or via an anonymous user ID if it is not. Its temporary nature though causes different needs for information storage and flow, as updates for single tickets occurs often. Hence the need for a separated module and a related database.

## Decision

We decided to pick Option 1. The increased concern for scalability, caused by event 1, points clearly to the listed pros of this option. Furthermore, the focus on modularity gives us easy maintainability, which is important for the TrIP owner. Performance in this case doesn't seem to be that central, as optimization of routes should be

precomputed and cached, given that timetables should not change often and that subscription have relatively long lasting periods. With our option, we still have the flexibility to assign more computational or data access resources to the module who needs them the most. We also include Option 3, as we think Single Tickets remain central to achieve the Usability required by stakeholders who are less familiar with technology such as smartphone apps or are travelling only occasionally.

## Consequences

### Positive Consequences

- **Enhanced Scalability:** Modular design facilitates easy scaling and integration of new features.
- **Improved Maintainability:** Simplifies updates and troubleshooting, allowing for technology-specific optimizations within each module.
- **Reduced System-wide Failure Risk:** Isolates failures to individual modules, enhancing overall system reliability.
- **Enhanced Usability for Single-Ticket buyers:** A specific module allows good handling of their needs with a high performance DB.

### Negative Consequences

- **Increased System Complexity:** Necessitates sophisticated coordination and integration, potentially introducing latency.
- **Higher Initial Costs:** Development and maintenance of separate modules may lead to higher initial and operational costs.
- **Data Consistency Challenges:** Requires robust synchronization mechanisms to ensure data consistency across modules.

The account management module is included in the functional view of the system, communicating with the tycoons to updated them on new subscriptions, with the database to keep the account data up to date and with the terminal to allow the purchase of new subscriptions and the verification via magnetic card or other means.

## 2.5 Decision 5: Concurrent payments / booking management

**Status**

Accepted

## Architectural Summary

| In the context of | managing high volumes of booking requests for a train, |
|---|---|
| **Facing** | the challenge of avoiding overbookings, |
| **To achieve** | high performance, minimal system overload and data consistency, prevention of passengers complaints, |
| **We considered** | Option 1: Lock the Seats Before Payment; Option 2: Lock the Seats After Payment, FCFS, Refuse Payments if Seat is Booked; Option 3: Real-time Seat Availability with Dynamic Allocation, Option 4: 2PC Protocol |
| **And decided for** | Option 1: Lock the Seats Before Payment, |
| **Because** | It directly solves the problem of overbookings and concurrent accesses, |
| **Accepting** | Increased System Complexity, potential for seats hoarding. |

## Concern

Ensuring a high level of usability and availability in the ticketing system is paramount. System operators and railroad tycoons aim to minimize customer service issues arising from passengers' frustrations with paying for unavailable routes.

- **User Story 15:** As an occasional passenger, I want the system to not allow me to pay for temporarily blocked routes, so that I don't have to go through customer service.,

- **User Story 26:** As a maintenance team member, I want the system to allow for temporary fare adjustments or bypass options during maintenance work so that we can minimize inconvenience to passengers.,

- **User Story 27:** As a network maintenance planner, I want the payment system to integrate with maintenance scheduling tools so that I can plan work with minimal disruption to the train service and revenue.,

## Context

This decision seeks to address challenges associated with quickly filling trains, especially during peak hours, to offer a seamless and fair ticket purchasing experience. Mitigating the risk of overbooking and enhancing passenger satisfaction are central goals. It's critical to efficiently manage simultaneous requests for the last available seats to ensure a positive experience for commuters and students who rely on timely and available transportation. It is relevant only for trains where seats must (or can) be booked.

**QA scenario**

The following outlines a Quality Attribute (QA) scenario focusing on the real-time seat availability and booking process during peak hours. This scenario is structured to evaluate the system's usability and performance under conditions of high demand.

**Source**  Passenger interacting with a payment terminal (or any other way of booking tickets, e.g. an app or a website).

**Stimulus**  Two passengers try to buy the same ticket.

**Artifact**  The *booking management module*, the *database* maintaining seat availability information and the *user interface* the passenger interacts with.

**Response** [*After the decision is taken*] Upon receiving the stimulus, the system's response is to immediately check the current availability of the selected seat, lock the seat for the passenger if it is available (thus preventing other bookings), and provide real-time feedback to the passenger regarding the seat's status (e.g., locked for purchase, already taken). If the seat is available and locked for the passenger, the system then proceeds with the payment process.

**Response Measure** [*After the decision is taken*] The effectiveness of the system's response is measured as follows:

- Percentage of purchases ending with passengers buying the same ticket. This can be measured by a customer service.

## Criteria

- *Scalability*: Minimize excessive requests to tycoon systems, avoiding system overload.

- Ensure high *performance* to prevent a negative user experience.

- *Functionality*: Prevent multiple passengers from paying for the same seat, ensuring fairness in ticket sales.

## Option 1: Lock the Seats Before Payment

We should maintain a database where info about scheduled trains is stored. We should also a booking management module that updates the database about booked seats or booking cancellations.

This database should be updated when a ticket has been bought at a terminal. The database should ask periodically for updates on schedules from the tycoon systems. When a user selects that they want to pay for a specific ticket, that ticket should be locked, so that no one else can buy it. If the payment is not ultimated, it can be unlocked. Note that it can still happen that due to maintenance, a train is cancelled last minute.

### Pros

- **Fairness** (Usability, Availability): Ensures that once a customer selects a ticket, it is reserved for them, preventing others from buying it.
- **Reduced System Load** (Performance, Efficiency): By locking seats before payment, it reduces the chances of multiple users attempting to pay for the same seat, thereby reducing system load.

### Cons

- **Potential for Seat Hoarding** (Usability, Efficiency): Customers might lock seats without completing the purchase, leading to temporarily reduced availability.
- **Increased Complexity** (Maintainability, Scalability): Managing locked seats, especially determining when to release them if payment is not completed, adds complexity.

### User stories

- Directly addresses **User Story 15** by preventing payments for unavailable seats.

- Indirectly supports **User Stories 26 and 27** by contributing to system usability and reducing overbooking, though it does not directly offer real-time updates or integration with maintenance scheduling.

## Option 2: Lock the Seats After Payment, FCFS, Refuse Payments if Seat is Booked

Seats are locked only after payment confirmation, adhering to a first-come, first-served (FCFS) approach. This method ensures that seats are sold to passengers who complete the payment process first, minimizing the potential for holding seats unnecessarily.

### Pros

- **Efficiency** (Performance): Minimizes the time seats are unnecessarily held, as they are only locked upon payment completion.
- **Simplicity** (Maintainability): Easier to implement and maintain than preemptive locking mechanisms.

**Cons**

- **User Experience** (Usability): Customers may go through the payment process only to find out the seat has been taken, leading to frustration.
- **Race Conditions** (Reliability): Higher risk of race conditions where multiple passengers complete payments for the last seat simultaneously.

**User stories**

- Aims to ensure fairness in seat allocation (**User Story 15**) by adhering to a first-come, first-served basis, reducing the risk of paying for unavailable seats.

- Does not directly address **User Stories 26 and 27**, but supports system performance and minimizes unnecessary seat holds.

## Option 3: Real-time Seat Availability with Dynamic Allocation

This option introduces a system for real-time tracking and dynamic allocation of seats. It involves continuous synchronization with tycoon systems to update seat availability instantly. The system could allow for a slight overbooking based on historical no-show rates, with safeguards in place to manage overbooked scenarios, such as offering alternative transportation options or compensation.

**Pros**

- **Real-time Updates** (Availability, Usability): Enhances user experience by providing immediate feedback on seat availability.
- **Adaptability** (Scalability, Reliability): Can dynamically adjust to changing conditions, like cancellations or no-shows.

**Cons**

- **Complex System Integration** (Maintainability, Operability): Requires continuous synchronization with external systems, increasing complexity.
- **Potential for Overbooking** (Usability, Reliability): While overbooking can be managed, it may lead to negative customer experiences.

**User stories**

- Directly solves **User Story 15**'s issue by ensuring passengers only pay for available seats and addresses **User Story 26** by providing real-time updates on train schedules.

- Supports **User Story 27** through dynamic allocation that can adjust to maintenance schedules, minimizing service disruptions.

## Option 4: 2PC Protocol

The Two-Phase Commit (2PC) protocol is a distributed transaction protocol that ensures all parts of a transaction across multiple systems either complete successfully or fail altogether. It is particularly useful in environments requiring strong consistency and atomicity, such as train terminal payment systems handling ticket purchases.

**Pros**

- **Atomicity and Consistency** (Reliability, Consistency): 2PC guarantees that a transaction across distributed components either fully commits or fully aborts, maintaining the integrity and consistency of the database.
- **Fault Tolerance** (Availability, Reliability): By ensuring that all components agree on a transaction's outcome, 2PC enhances the system's ability to recover from partial failures without losing data integrity.
- **Coordination** (Integrity, Consistency): 2PC effectively coordinates complex transactions across multiple systems, ensuring that all parts of the transaction are synchronized.

**Cons**

- **Performance Overhead** (Performance, Efficiency): The two-phase nature of the protocol can introduce latency, as it requires all participants to lock resources and wait for global commit or abort decisions.
- **Resource Locking** (Availability, Scalability): 2PC requires resources to be locked during the transaction, which can decrease system availability and limit scalability due to locking contention.
- **Complexity** (Maintainability, Operability): Implementing and maintaining a 2PC system introduces complexity, requiring robust failure and recovery mechanisms, and can complicate system operations and maintenance.
- **Risk of Blocking** (Availability): In cases where the coordinator fails after initiating the transaction but before completing it, participants can be left in a blocking state, waiting indefinitely for a decision and thus affecting system availability.

**User stories**

- Indirectly supports **User Story 15** by ensuring transactional integrity, which is foundational for reliable seat booking but does not directly address usability concerns.

- Does not directly address **User Stories 26 and 27**, but by ensuring consistent and reliable transactions, it lays the groundwork for future enhancements that could.

## Decision

In light of the trade-off analysis performed for each Option listed above, we decided to take Option 1. This Option ensures a solution to **User Story 15**, by effectively preventing two users to buy the same tickets. It is also good for reliability, a focus of the tycoons and for the usability required by the passengers, as they don't have to restart the booking procedure in case of failed payment. It seems to satisfy all the listed criteria. Other options can lead to overbooking, which is unwanted given the concerns of passengers and tycoons. Option 4 can be useful if the system is decentralized, but introduces unwanted complexity.

## Consequences

**Positive Consequences:**

- Improved passenger experience through fair and efficient seat allocation.

- Reduced customer service issues related to overbooking and ticket availability.

- Enhanced system resilience against peak demand scenarios.

**Negative Consequences:**

- Potential increase in system complexity and operational costs.

- Challenges in accurately forecasting demand for dynamic seat allocation or overbooking strategies.

- Possible passenger dissatisfaction in cases of overbooking or changes in train schedules.

## 2.6 Decision 6: Database scope and organization

### Status

Review.

### Architectural Summary

| In the context of | choosing which data should be stored in the TrIP system, |
|---|---|
| **Facing** | the challenge of ensuring data security and high performance, |
| **To achieve** | high performance in data retrieval, easy integrability with tycoons' systems, |
| **We considered** | Option 1: Database-per-service pattern; Option 2: Unified Centralized Database System; Option 3: Database per tycoon, Option 4: Hybrid cloud storage |
| **And decided for** | Option 1: Database-per-service pattern, Option 4: Hybrid cloud storage |
| **Because** | It allows us to achieve high availabiliy and maintenability, |
| **Accepting** | Increased System Complexity, higher costs. |

### Context

We need to answer the following questions:

- Which info do we keep in the database?

- Do we need one or many databases?

- How do we handle data protection?

- How do we handle data update from tycoons/station management? (maybe for another decision?)

### Concern

We need to keep some information available from our system, but security of the data (especially payment and account data) is crucial for the government and for the passengers concerns. We also need high performance in giving routes choices to passengers and quick payments.

The following user stories are particularly relevant to the decision on the database scope, emphasizing the need for comprehensive management of subscriptions, security, and system integration:

- **User Story 1:** As a frequent traveler, I want to subscribe to a comprehensive monthly pass that includes all three networks so that I can save money on my regular commutes.,
- **User Story 2:** As a passenger, I want to be able to check the balance of my multi-network travel card online so that I can easily manage my travel expenses.,
- **User Story 3:** As a passenger with a subscription, I want to receive automatic notifications about my subscription renewal and any discounts or promotions available so that I can take advantage of cost savings.,
- **User Story 4:** As a tech-savvy passenger, I want to use a mobile app to manage my subscriptions, make payments, and receive digital tickets so that I can have a paperless and convenient travel experience.,
- **User Story 5:** As a passenger interested in sustainability, I want the system to track my travel carbon footprint and offer carbon offset options so that I can make environmentally responsible travel choices.,
- **User Story 6:** As a passenger with mobility challenges, I want the payment system to provide information on accessible services and allow for easy purchase of accessible seating across all networks so that I can travel comfortably and safely.,
- **User Story 16:** As a passenger, I want to be able to purchase a single ticket that allows me to travel across all train networks so that I can travel seamlessly without needing to buy separate tickets for each network.,
- **User Story 18:** As a government regulator, I want the payment system to adhere to data protection and financial transaction security standards so that passengers' personal and financial information is secure.,
- **User Story 23:** As a tycoon, I want the payment system to integrate with my existing infrastructure with minimal disruption so that I can maintain high service levels during the transition.,
- **User Story 27:** As a network maintenance planner, I want the payment system to integrate with maintenance scheduling tools so that I can plan work with minimal disruption to the train service and revenue.,

## Criteria

- *Performance and availability*: Quick and reliable data retrieval.

- *Mainteinability*: clear data organization and division of responsibilities across system elements.

- *Security*: Protection of sensitive data.

## Option 1: Database-per-service pattern

We try to have one database for each important set of information that the system needs to know, employing the database-per-service pattern. We try to stick to the Separation of Concerns principle, so that each module only has access to the data it scricly needs to operate. We divide the following:

- A database containing the timetable, the prices of seats and the bookings.

- A database containing the optimized routes cached after being calculated by the Routes Optimization Module.

- A database containing info about the accounts and their subscriptions.

- A database recording payments that have been done, which should be accessed in case for example of complaints.

**Functional View - Databases**
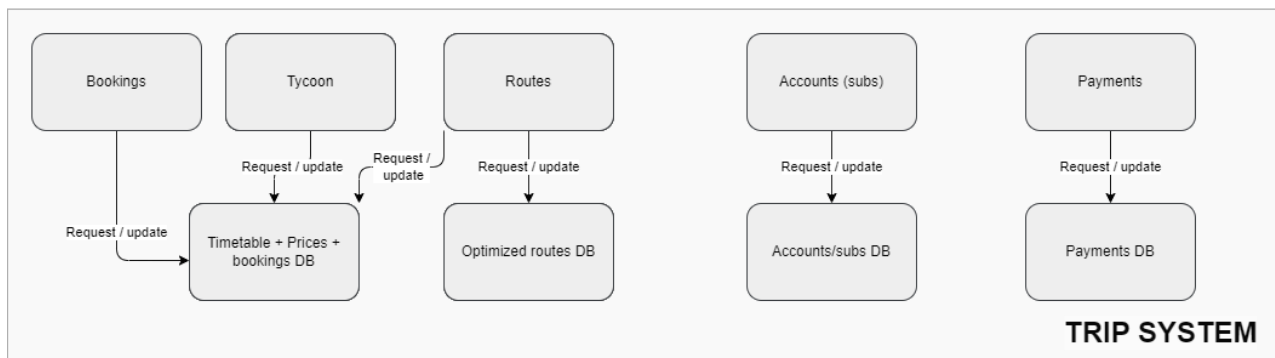


Figure 4: Division of databases and their interaction with modules or stakeholders.

**Pros**

- **Enhanced Security** (Data Protection, Privacy): By segregating data across multiple databases, sensitive information is better protected, and access can be tightly controlled on a need-to-know basis.
- **Specialized Optimization** (Performance, Efficiency): Dedicated databases allow for optimization specific to their function, such as faster queries for timetable and booking data versus complex route optimization calculations.

**Cons**

- **Increased Maintenance Overhead** (Maintainability, Complexity): Managing multiple databases adds complexity to the system's architecture, requiring more resources for maintenance and potentially higher costs.
- **Data Synchronization Challenges** (Reliability, Consistency): Ensuring data consistency across different databases can be challenging, especially in real-time, and may affect the system's overall reliability.

## Option 2: Unified Centralized Database System

This option proposes a centralized database architecture that consolidates all necessary information into a single, unified database system. It incorporates robust access control layers to manage data access based on module or user roles, ensuring that each part of the system accesses only the data it needs for operation. This model simplifies data management, enhances security through centralized control mechanisms, and facilitates easier updates and integrations. This model can be improved with having accounts database as a separate component, and keeping the rest of the databases central. In this way accounts data will be secure, and the rest of the systems will access accounts data anonymously via ids. Manage permissions for eacsh tycoon, on which info they can access (they shouldn't be able to connect name to bank account). In this way we can keep everything in the same place, but each tycoon api will have specified access protocols. Tycoons can only update timetables and prices.

We can keep bookings data a DaaS, since it needs to be updated regularly and concurrently. Rest can be a server based database. Since we can cache routes we don't want cloud services for these.

### Pros

- **Simplified Data Management** (Maintainability, Efficiency): Centralizing data storage simplifies the architecture by reducing the number of systems to manage, making it easier to maintain and update the database.
- **Improved Data Consistency** (Reliability, Integrity): A unified database ensures that all modules access the most current and consistent data, reducing the risk of discrepancies and errors.
- **Enhanced Integration Capability** (Scalability, Interoperability): With all data in one place, integrating new features, modules, or external systems becomes more straightforward, promoting scalability and interoperability.

### Cons

- **Risk of a Single Point of Failure** (Reliability, Availability): Centralizing data creates a single point of failure, which could potentially lead to system-wide outages affecting all functionalities if the database goes down.
- **Scalability Concerns** (Performance, Scalability): As the system grows, a centralized database might struggle with performance issues due to the increasing volume of data and concurrent access requests.
- **Complexity in Ensuring Data Protection** (Security, Privacy): Protecting a large, centralized repository of sensitive information poses significant challenges, requiring robust security measures to prevent unauthorized access and data breaches.

## Option 3: Database per tycoon

### Pros

- **Tycoon specific features**
- **Tycoon access permissions are automatically managed**

### Cons

- **Hard to manage the database**
- **Shared data is potentially doubled**

## Option 4: Hybrid cloud storage

Crucial data stored in cloud (timetables, and prices and bookings). Non-crucial data (accounts, payments) gets stored in non-cloud database one per tycoon, such that they can only access their own customer data.

### Pros

- **Tycoon specific access for accounts data**
- **Secure and qucik access to data crucial data**
- **Cheap storage for non-crucial data**

### Cons

- **Can get expensive**
- **Implementation of two different database types and their interactions setup**

## Decision

We choose Options 1 and 4, using a Hybrid cloud storage System for the deployment, but separate databases accessible by one related service. The goal is to achive high Availability, which is crucial after Event 2, which requirees less stringent Maintainability and cost requirements. Maintainability and performance are increased by choosing database-per-service instead of a single database. Cloud solutions can be expensive for data storage.

## Consequences

**Positive Consequences:**

- **database as a service**: Single point of failure is reduced, scalable.

**Negative Consequences:**

- **Tycoon permission management** : More complexity, we need to restrict access for each tycoon.

- **Single point of failure** : If database fails, everything fails.(bad for availablity)[maybe a solution is backup database].

- **database as a service**: Expensive.

## 2.7 Decision 7: Customer Service

**Status**

Review.

### Architectural Summary

| In the context of | Providing customer support to passengers, |
|---|---|
| **Facing** | The passengers' and customer service operators concern for quick information retrieval from the system, |
| **To achieve** | Timely and accurate, secure information retrieval and communication, |
| **We considered** | Option 1: Direct Access to Live Data; Option 2: Periodic Data Sync to a Dedicated Customer Service Database; Option 3: On-Demand Data Retrieval via Secure API; Option 4: Automated Reporting System |
| **And decided for** | Option 3: On-Demand Data Retrieval via Secure API |
| **Because** | It ensures security and good flexibility to integrate with changes in the customer service platform, |
| **Accepting** | Possible latency, an additional layer of complexity. |

### Concern

The primary concern is to ensure that customer service representatives have access to accurate and timely information to address passenger queries and resolve issues efficiently, without compromising data privacy.

Related user stories are listed here:

- **User Story 18:** As a government regulator, I want the payment system to adhere to data protection and financial transaction security standards so that passengers' personal and financial information is secure.,

- **User Story 26:** As a maintenance team member, I want the system to allow for temporary fare adjustments or bypass options during maintenance work so that we can minimize inconvenience to passengers.,

### Context

This decision outlines the strategy for communication between the TrIP system and customer service teams to facilitate rapid and effective resolution of customer issues. Customer service teams require real-time access to passenger data, ticketing information, and system status to provide informed support. The chosen communication strategy must balance the need for information accessibility with system security and data privacy regulations.

### Criteria

- *Functionality and performance*: Timeliness and accuracy of information communicated.

- *Security*: Data privacy and security compliance.

- *Usability*: Ease of access for customer service representatives.

- *Mainteinability*: Minimization of system complexity and maintenance.

- *Integrability*: Integration with existing customer service platforms.

- *Cost efficiency*: Cost-effectiveness of the communication solution.

### Option 1: Direct Access to Live Data

Grant customer service representatives direct access to the live operational database with appropriate read-only permissions and privacy safeguards in place.

**Pros**

- Immediate access to data allows for quick customer service responses.

**Cons**

- Direct access to live data could pose security risks if not managed correctly.

## Option 2: Periodic Data Sync to a Dedicated Customer Service Database

Regularly synchronize relevant data from the operational database to a separate customer service database designed for query efficiency and tailored access control.

### Pros

- Data syncing provides a stable environment tailored for customer service needs.

### Cons

- Data syncing could lead to delays in information relay if not frequent enough.

## Option 3: On-Demand Data Retrieval via Secure API

Implement a secure API that allows customer service representatives to retrieve necessary data on-demand while maintaining strict access controls and audit trails.

### Pros

- Secure API ensures data privacy and minimizes unnecessary data exposure.

### Cons

- On-demand retrieval may introduce latency and requires robust API management.

## Option 4: Automated Reporting System

Develop an automated reporting system that provides customer service representatives with pre-defined reports and dashboards, reducing the need for direct data access.

### Pros

- Automated reports streamline the information delivery process.

### Cons

- Automated reporting may not cover all ad-hoc queries from customer service representatives.

## Decision

Option 3 is chosen: not enough requests to justify the burden of an additional database. Option 4 requires too much work from us. Option 1 doesn't seem secure enough.

## Consequences

**Positive Consequences:**

- **Security and Privacy:** The secure API maintains strict access controls and audit trails, enhancing the protection of sensitive passenger data and ensuring compliance with data privacy regulations.
- **Flexibility:** Allows for flexible, on-demand data retrieval, which can be easily adapted or expanded to meet evolving requirements, providing a scalable solution for future needs.
- **Integration Capabilities:** Facilitates easier integration with existing or future customer service platforms, making the option highly scalable and versatile for a range of services.

**Negative Consequences:**

- **Latency and Performance:** The on-demand nature of data retrieval may introduce latency, requiring robust API management and performance optimization to ensure timely customer service responses.
- **Complexity in API Management:** Managing the API adds a layer of complexity, including version control, access management, and security updates, which necessitates dedicated resources.

- **Dependence on External Systems:** Reliance on external services or platforms for the API functionality can pose risks related to their availability and performance, necessitating contingency planning for high availability and redundancy.

## 2.8 Decision 8: Interaction with ticket scanners

### Status

Open

### Architectural Summary

| In the context of | turnstile interaction with the TRiP system, |
|---|---|
| **Facing** | the passengers' request for a seamless travel experience and tycoon need to admit only authorized passengers in their stations, |
| **To achieve** | high availability, security in sensitive data communication, many options adapted to different passengers needs, |
| **We considered** | Option 1: Add a ticket checker and a user state modules to authorize passengers; Option 2: Allow fillable cards, account cards and tickets, not bank cards; |
| **And decided for** | Option 1: Add a ticket checker and a user state modules to authorize passengers; |
| **Because** | Not allowing bank cards would make us lag behind with respect to modern train payment systems that allow high usability for tech-savy passengers, |
| **Accepting** | The need for an offline mode in case the network fails, possible latency. |

### Concern

Passengers want a seamless travel experience, without being mistakenly blocked at the turnstiles. Tycoons want to admit only paying passengers in trains, avoiding controllers costs. Turnstiles need to communicate with the system to ensure correct and quick verification of the requisites to be allowed in the station.

### User Stories

The decision on how turnstiles interact with the payment system is integral to ensuring a seamless travel experience and securing access to train areas. Below, we detail specific user stories connected to this architectural decision, highlighting the requirements and expectations from various stakeholders.

1. **User Story 1:** As a frequent traveler, I want to subscribe to a comprehensive monthly pass that includes all three networks so that I can save money on my regular commutes.,
2. **User Story 2:** As a passenger, I want to be able to check the balance of my multi-network travel card online so that I can easily manage my travel expenses.,
3. **User Story 16:** As a passenger, I want to be able to purchase a single ticket that allows me to travel across all train networks so that I can travel seamlessly without needing to buy separate tickets for each network.,
4. **User Story 26:** As a maintenance team member, I want the system to allow for temporary fare adjustments or bypass options during maintenance work so that we can minimize inconvenience to passengers.,
5. **User Story 27:** As a network maintenance planner, I want the payment system to integrate with maintenance scheduling tools so that I can plan work with minimal disruption to the train service and revenue.,
6. **User Story 23:** As a tycoon, I want the payment system to integrate with my existing infrastructure with minimal disruption so that I can maintain high service levels during the transition.,

These user stories collectively highlight the diverse requirements for the turnstile and payment system interaction, from seamless access and real-time information integration to flexibility in handling various ticketing formats and system updates. The problem can be made more abstract, as the interaction with the turnstile can be similar to the following interactions:

- interaction with a controller who scans the ticket or any proof of subscription.

- interaction with a scanner onboard a bus.

## Context

Most stations, expecially the big ones, have turnstiles to avoid people without tickets to enter the trains area. The turnstiles should communicate with the system to ensure this. A decision should be taken on who to admit in the station. This logic could be provided by the station manager or by the tycoons, or even by TrIP managment. How do the passenger input its ticket or subscription or prepaid card to the turnstile? How does the turnstile communicate with the system to verify if the passenger has the right to enter the station?

## Criteria

- *Usability and functionality*: Allow passengers to scan tickets or multiplatform travel cards (if they exists) at turnstiles.

- *Safety*: Don't allow people without the appropriate tickets or subscriptions or cards.

- *Availability and performance*: Let passenger quickly open turnstiles when they are authorized.

- *Functionality*: Possibly allow people to scan when they enter and when they exit and calculate a fare (this might depend on agreements between tycoons).

- *Functionality*: Possibly communicate station usage data to the system.

- *Functionality*: Possibly allow credit/debit card scan.

- *Functionality*: Allow balance check for charged cards.

## Option 1: Add a ticket checker and a user state module to authorize passengers

After scanning a ticket or badge, the user state should be updated, in order to calculate the fare price for fillable card holders or for the scanning of bank cards (debit credit). The ticket or badge should be checked by the single ticket checker module or by the account management to insure consistency. In case of the scanning of bank cards, the user state module needs to communicate with payment management, in order to proceed with the actual payment. The scanning of bank cards and fillable cards create the needs for a calculation of appropriate revenue share between different tycoon. This should be the topic of a future decision or might be incorporated in this one.

## Functional View - Scanning Procedure
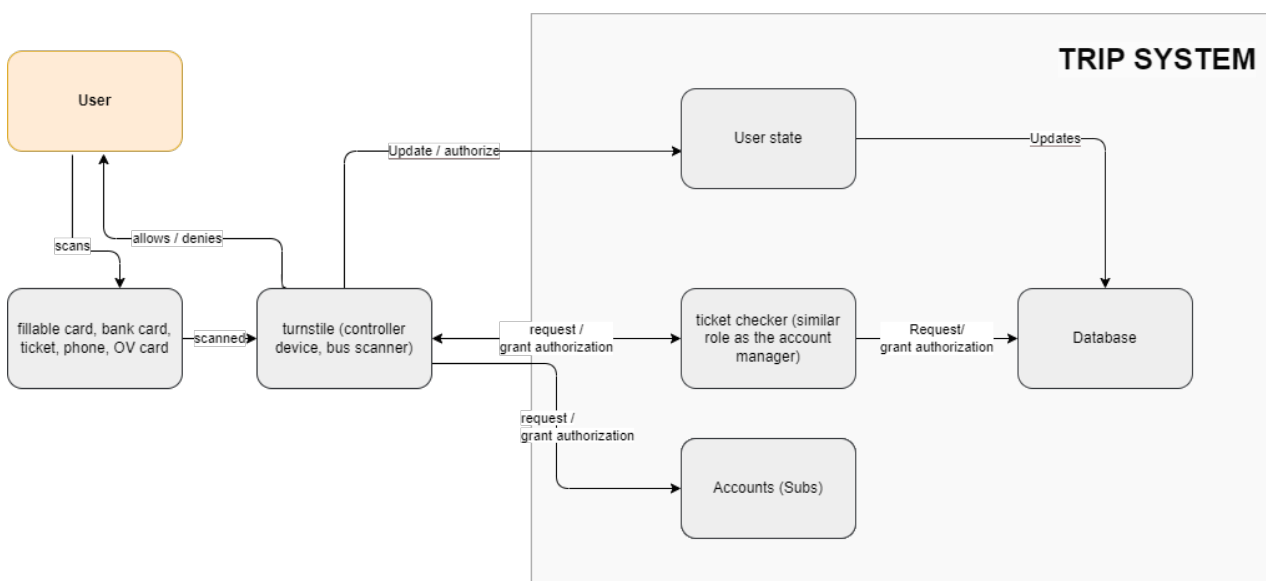


Figure 5: Interaction with a ticket scanner.

29

## Option 2: Allow fillable cards, account cards and tickets, not bank cards

Not allowing bank cards at turnstiles simplifies the job of turnstiles. Also, how can a controller check if the traveller have scanned a bank card when entering the turnstile? We can have the turnstile print a ticket with a barcode or QR code. This way all the relevent information is contained in the card or ticket and no on-line check is needed.

## Decision

We choose Option 1, as refusing bank cards checking would hinder usability for the passengers. It would make us lag behind with respect to other train payments systems. The security risk that this poses can be mitigated by applying an encryption tactic (this will be explained in more detail in the information viewpoint). An offline mode needs to be detailed to face Event 2 (possible network disruptions in stations), this will be explained in the concurrency viewpoint.

## Positive Consequences

- Enhances security by ensuring only authorized passengers can access services.

- Improves passenger experience with seamless turnstile interaction.

- Supports real-time updates for station managers, enhancing operational efficiency.

- Integrates with modern payment systems, offering convenience to tech-savvy passengers.

- Provides for an offline mode, ensuring system reliability in case of network failures.

## Negative Consequences

- Introduces system complexity and potential integration challenges.

- Creates dependency on external payment systems, raising operational risks.

- Incurs significant operational costs for implementation and maintenance.

- Raises data privacy and security concerns due to sensitive information handling.

- Complicates the revenue-sharing model among tycoons, requiring detailed negotiations.

## 2.9   Decision 9: Account management and authentication

### Status

Review

### Architectural Summary

| In the context of | letting passengers authenticate to their accounts to handle subscriptions, |
|---|---|
| Facing | the passengers' concern for user-friendlyness and the tycoon request for integration between their subscriptions systems, also ensuring sensitive data privacy, |
| To achieve | security, usability for passengers and integrabiliy with tycoons systems, |
| We considered | Option 1: Anonymous card; Option 2: Bank credit card; Option 3: Phone app; Option 4: Nominal card; Option 5: NFC reader/Google Wallet |
| And decided for | a hybrid approach where multiple authentication methods are allowed, both nominal (like a dedicated card) and anomymous (like a fillable card); |
| Because | it ensures high usability for many kinds of passengers, only two readers are needed (NFC and QR) at turnstiles, which keeps costs controlled, security is ensured by tokenization, |
| Accepting | Higer system complexity, more complicated development and maintenance. |

### Concern

Ensuring a secure, user-friendly, and efficient process for account creation, management, and authentication for a variety of users, while maintaining data privacy and system integrity.

Related user stories are listed below:

- **User Story 1:** As a frequent traveler, I want to subscribe to a comprehensive monthly pass that includes all three networks so that I can save money on my regular commutes.

- **User Story 4:** As a tech-savvy passenger, I want to use a mobile app to manage my subscriptions, make payments, and receive digital tickets so that I can have a paperless and convenient travel experience.

- **User Story 12:** As a commuting worker/student, I want a single transaction to get all the subscriptions I need, so that I can travel from home to work/school every day.

- **User Story 18:** As a government regulator, I want the payment system to adhere to data protection and financial transaction security standards so that passengers' personal and financial information is secure.

- **User Story 26:** As a maintenance team member, I want the system to allow for temporary fare adjustments or bypass options during maintenance work so that we can minimize inconvenience to passengers.

### Context

The TrIP system requires a flexible and secure method for user authentication that accommodates various levels of user interaction and convenience. The chosen solution must integrate seamlessly with the existing ticketing and payment infrastructure and support a range of devices and technologies used by passengers.

### Criteria

- *Security*: Secure storage and handling of personal and financial data.

- *Usability*: Ease of account creation and management for passengers.

- *Integrability*: Seamless integration with existing turnstile and payment systems.

- *Flexibility*: Flexibility to support different methods of authentication, including cards and digital wallets.

- *Availability*: High availability and reliability of the authentication system.

## Option 1: Anonymous card

- **Pro:** Ensures passenger privacy and quick adoption for casual users.

- **Con:** Limited capabilities for account management and tracking passenger history.

## Option 2: Bank credit card

- **Pro:** Streamlines payment process by integrating with existing financial systems.

- **Con:** Relies on external systems, which may pose integration challenges and dependency risks.

## Option 3: Phone app

- **Pro:** Provides a versatile platform for account management, payments, and ticket validation via QR codes or NFC.

- **Con:** Requires smartphone access, possibly excluding certain passengers demographics.

## Option 4: Nominal card

- **Pro:** Offers a physical, personalized token for account access and ticket validation.

- **Con:** May introduce additional costs for production and distribution of cards.

## Option 5: NFC reader/Google Wallet

- **Pro:** Leverages existing NFC technology in smartphones for easy tap-and-go access at turnstiles.

- **Con:** Implementation cost and required NFC-capable turnstiles may be high.

## Decision

The decision is to implement a hybrid approach, integrating a phone app (with NFC capabilities and linked to bank accounts) for regular users, along with an anonymous fillable card for tourists and a physical nominal card for those who prefer or require a physical token. The hybrid approach balances convenience with coverage for all passenger types. A tokenization service can be implemented, so that no sensitive data is stored, but only tokens representing it, which transfer the security risk and the compliance requirements to specialized companies. This will be explained in more detailed in the Information Viewpoint.

## Consequences

**Positive Consequences:**

- Provides a comprehensive solution covering the needs of various passenger groups, increasing system accessibility and passenger satisfaction.

- Enhances security by offering multiple authentication methods, each with its own set of security protocols.

- Encourages digital transformation and supports a move towards a more contactless, efficient user experience.

**Negative Consequences:**

- May increase complexity and cost of the system, both in development and maintenance.

- The need to support multiple authentication methods can complicate infrastructure and operational processes.

- Potential resistance from passengers who are less technologically adept or prefer traditional methods.

This decision supports the TrIP system's objective to provide a secure, convenient, and inclusive environment for all types of passengers while recognizing the need to manage operational and development complexities.

## 2.10 Decision 10: Data update from tycoons or station management

### Status

Accepted.

### Architectural Summary

| In the context of | allowing data update (like timetables and fare prices) from tycoons, |
|---|---|
| Facing | the need to quickly allow addition of new tycoons, fair competition between tycoons, passengers' request of correct information; |
| To achieve | tycoons' usability, availability for passengers, system's mainteinability, |
| We considered | Option 1: Tycoon API for interactions with databases; Option 2: Real time information requests to tycoon systems and tycoon-specific API |
| And decided for | Option 1: Tycoon API for interactions with databases; |
| Because | Convers Event 1 specifically, ensures data consistency across tycoons-specific timetables/price, improves mainteinability of the system, |
| Accepting | Potential resistance from some tycoons, API management, data management overhead. |

### Concern

The system needs to receive information and updates from the tycoons. This can be for instance:

- train or buses time tables;

- static data like stations ownership and connections (static meaning that it changes less often than other information);

- last-minute updates like disruptions, delays, interruptions.

The information above are necessary for the following quality attributes:

- *reliability* of the system required by the tycoons (priority 2), as passengers needs to always be able to pay for their travels (hence updated timetables are important);

- *usability* of the system for the passengers (priority 2), who wants as few actions on their own initiative as possible;

The system also need to give acces to some information to the tycoon, with proper care to not expose information from other tycoons:

- payment received by subscribers;

- bought tickets;

- booking information;

- data from the turnstiles scanners.

Scalability here is an important concern, as this data needs to be provided, requested and stored by the system for different tycoons. After Event 1, the priority for *Scalability* has been set to two, so the system needs to be flexible enough to easily access new tycoons also from different kind of transport.

Also *security* and *privacy* here are paramount, as only the strictly necessary and allowed information can be exposed.

### User stories

- **User Story 26:** As a maintenance team member, I want the system to allow for temporary fare adjustments or bypass options during maintenance work so that we can minimize inconvenience to passengers.

- **User Story 35:** As a new tycoon entering the market, I want to easily integrate my services with the TrIP system, so that passengers can quickly benefit from additional travel options.

- **User Story 36:** As a system administrator, I want the ability to add and configure services from new tycoons without significant downtime, so that the system remains up-to-date and competitive.

- **User Story 37:** As a passenger, I want to see new transportation options as soon as they become available, so that I can plan my trips with the most comprehensive information.

- **User Story 38:** As a tycoon, I want my data (such as timetables, prices, and updates) to be accurately and promptly reflected in the system.

## Context

The systems keeps data stored on multiple databases. Some of them needs to give some degree of access to the tycoons and the customer service of the TrIP system:

- Tickets database;

- Payments database;

- Accounts and subscriptions database;

Others needs to let the tycoon feed them, such as the Timetable databases, which holds information about train schedules, prices and bookings.

### QA Scenario

Consider the scenario of adding/removing a tycoon, adding/removing routes and stations.

## Criteria

- *Availability*: Data correctness in the system, expecially after Event 2.

- *Usability*: Ease of use from the tycoons to update the system and get data for analytics.

- *Performance* and *availability*: data should flow into the system with reasonable speed.

- *Scalability and mainteinability*: Adding new tycoons should be easy.

## Option 1: Tycoon API for interactions with databases

We can add a layer between the TrIP system and the Tycoons, as an API which gives a set of querying rights to tycoons to export the data they need and forces them a format of data to be provided.

Forcing tycoons to a certain format is possible because both bus and train system can be abstracted to a graph, where nodes are stations and edges are connections. They both have times for connections and prices to submit, together with bookings availabilities.

### Pros

- **Scalability**: by forcing a format of queries and data submission, we abstract away from the specifics of a way tycoons provide their connection services.
- **Maintainability**: we let our system have its own independent and unique data representation, ensuring a common data format.
- **Performance**: Routes optimization can be precompute with timetables and only updated when new information come from the tycoond.

### Cons

- **Usability** for the tycoon: Interfacing with an API needs some work from the tycoons IT department, to ensure the TrIP system is properly updated.
- **Operational cost**: storing lots of information on the TrIP system requires the management of multiple databases.

## Option 2: Real time information requests to tycoon systems and tycoon-specific API

Also asking data from tycoons periodically from the system is a possibility. In this case the TrIP system is responsible for the data querying and feeding, thus it should be able to interact with each tycoon system with a specific API.

**Pros**

- **Usability**: The system can provide almost real-time to users, only limited by the technical capabilities of tycoons systems.
- **Operational costs**: The system doesn't need to store as much information, as it can pass it to the tycoons, let them store it and then cancel it.

**Cons**

- **Maintainability** and **Scalability**: This requre a lot of implementation every time a new tycoon enters.

## Decision

As the importance of scalability has been increased by event 1, Option 1 seems the most natural. Furthermore, operational cost has reduced importance after event 2, which limits the cons of option 1. It ensures high integrability of the system with the tycoons and customer services systems. Also availability can be improved with option 1, as we cannot ensure high availability of tycoon system, while we can deploy tactics to ensure our own databases availabiliy. The specific choice of tactics will be outlined in the deployment view.

## Consequences

**Positive Consequences:**

- Adding new tycoons would be easy for the TrIP system.

- Standardized data is exchanged and managed.

- Load management efficiency: the TrIP system is responsible for its own load management, instead of relying on tycoons systems.

- High accuracy of data thanks to standardization.

- Tycoons can potentially share analytical tools, having all the same interface.

- API acts as a gatekeeper, to keep tycoons from accessing to unwanted information. This should be reflected on future decisions meant at defining specific tactics.

**Negative Consequences:**

- Potential resistance from some tycoons (we could mitigate by providing technical support);

- API stability: updates to our API might require changes from many tycoons, therefore the API will be by definition difficult to modify. Also here a proper technical support could be provided to mitigate the issue.

- Data management overhead: we need data storage and backup and security concerns. Storing payment data can be done in an anonymized way using tokenization services.

## 2.11 Decision 11: Disruptions and Route Updates

### Status

Open

### Architectural Summary

| In the context of | Managing disruptions effectively within the TrIP system. |
|---|---|
| **Facing** | The need to minimize inconvenience for passengers during disruptions. |
| **To achieve** | A balance between rapid response and clear communication with passengers. |
| **We considered** | 1. Real-Time Alert System, <br> 2. Manual Intervention Protocol, <br> 3. Threshold-Based Re-Optimization, <br> 4. Continuous Optimization. |
| **And decided for** | Option 3: Threshold-Based Re-Optimization. |
| **Because** | It optimally balances operational efficiency and passenger communication, focusing resources on significant disruptions. |
| **Accepting** | The potential oversight of minor delays and the reliance on terminal-based information. |

### Concern

The main concern is to ensure minimal inconvenience to passengers during train disruptions while maintaining transparent communication and providing alternative solutions.

Related user stories are listed below:

- **User Story 26:** As a maintenance team member, I want the system to allow for temporary fare adjustments or bypass options during maintenance work so that we can minimize inconvenience to passengers.

### Context

Train disruptions can occur due to various reasons such as maintenance issues, accidents, or natural events. The system needs to be able to quickly respond to such incidents, inform affected passengers, and offer alternatives to ensure continued service. This is important for our system as a consequence of the choice to handle route optimization within the system.

### Criteria

- Rapid detection and response to disruptions.

- Clear and timely communication with passengers.

- Provision of alternative transport options.

- Integration with existing operational and communication systems.

- Minimization of negative impact on passenger experience.

- Compliance with safety and regulatory standards.

### Option 1: Real-Time Alert System

**Pros**

- Quickly informs passengers about disruptions.

**Cons**

- Requires passengers to actively seek updates.

## Option 2: Manual Intervention Protocol

**Pros**

- Personalized assistance to passengers for rebooking and advice.

**Cons**

- Complex to implement and integrate with existing systems.

## Option 3: Threshold-Based Re-Optimization

**Pros**

- Focuses on significant disruptions, optimizing system and passenger resources.
- Reduces the number of unnecessary passenger notifications for minor issues.

**Cons**

- Minor delays may not trigger system responses, potentially accumulating unnoticed.
- Terminal-based information may not effectively reach all passengers.

## Option 4: Continuous Optimization

**Pros**

- Offers constant adaptability to real-time conditions, potentially enhancing system responsiveness.

**Cons**

- Could result in frequent, possibly confusing updates for passengers.
- May demand significant computational resources, affecting system efficiency.

## Decision

We choose Option 3: Threshold-Based Re-Optimization for its strategic focus on significant disruptions. This decision is made recognizing that while minor disruptions may be overlooked, the emphasis on substantial delays aligns with our goal of efficiently managing resources and maintaining a high-quality passenger experience.

### Positive Consequences

- Efficient resource allocation by focusing on significant disruptions ensures the system's responsiveness to passenger needs during major incidents.
- Reduction in passenger notification fatigue by limiting communications to significant events, thereby enhancing the relevance and impact of messages received.
- Improved system performance and cost efficiency by avoiding unnecessary re-optimization processes for minor disruptions.

### Negative Consequences

- Potential for minor disruptions to accumulate and impact passenger experience if they are not addressed due to falling below the set threshold.
- Risk of insufficient communication if passengers are not near terminals and thus may miss critical information about disruptions and re-optimizations.
- Challenges in setting an appropriate threshold that accurately distinguishes between minor and significant disruptions, requiring continuous evaluation and adjustment.

## 2.12   Decision 12: Serverless vs. Servers for Calculations

### Status

Open

### Architectural Summary

| In the context of | choosing to deploy TrIP system on cloud or on physical servers, |
|---|---|
| Facing | cost-effectiveness required by the TrIP owner, scalability and performance needs, |
| To achieve | high availability of the system together with security of passengers data, |
| We considered | Option 1: Serverless architecture; Option 2: Dedicated Servers; Option 3: Hybrid Approach |
| And decided for | Option 3: Hybrid Approach |
| Because | it ensures security where it is needed and cost-effectiveness for the rest of the system, |
| Accepting | Higer operation costs and more involved initial setup and mainentance. |

### Concern

The primary concern is to select a computational architecture that balances scalability, cost, performance, and data privacy for processing passenger data and transactional information.

Related user stories are listed here:

- **User Story 29:** As the TrIP owner, I want the system to operate with the lowest possible operational costs, so that it remains cost-effective over time.

- **User Story 30:** As the TrIP owner, I want the system maintenance efforts to be easily scalable, so that costs do not increase disproportionately as we expand.

- **User Story 31:** As the TrIP owner, I want routine maintenance tasks to be automated, so that manual labor and operational costs are reduced.

- **User Story 18:** As a government regulator, I want the payment system to adhere to data protection and financial transaction security standards so that passengers' personal and financial information is secure.

### Context

The computational backbone of the TrIP system must handle variable workloads efficiently, especially during peak hours when route calculations and payment processing are at their highest demand. Additionally, the system must maintain data privacy and adhere to regulatory compliance.

### Criteria

- *Scalability* to handle peak and off-peak loads.

- *Cost efficiency*, including operational and maintenance costs.

- *Performance* in terms of latency and throughput.

- *Security*: Data privacy and control.

- Compliance with data protection and privacy laws.

- *Maintainability*: Ease of maintenance and updates.

### Option 1: Serverless architecture

Adopting a serverless architecture where the service provider dynamically manages the allocation of machine resources.

**Pros**

- Cost efficiency during low usage.

- No need for server maintenance.

- Automatic scaling.

**Cons**

- Potential for increased latency.

- Less control over data.

- Possible security concerns.

## Option 2: Dedicated Servers

Using dedicated servers, either on-premises or hosted, to handle all computations.

**Pros**

- Greater control over data.

- Potentially better performance.

- Consistent availability.

**Cons**

- Higher upfront costs.

- Requires dedicated IT staff for maintenance.

- Might be underutilized during off-peak times.

## Option 3: Hybrid Approach

Implementing a hybrid system that uses a combination of serverless architecture for less sensitive and highly variable workloads, and dedicated servers for more predictable workloads and data-intensive tasks requiring stringent privacy controls.

**Pros**

- Balances the benefits of both serverless and dedicated servers.

- Provides scalability while maintaining data privacy for sensitive operations.

**Cons**

- Increased complexity in managing two different environments.

- Potential for higher operational costs.

## Decision

As the focus on security have increased after Event 4 and the TrIP owner is more prone on spending, we pick option 3. This way, all the privacy-critical operations and data storages will be done in private clouds, while operations and storages of public data (such as the timetables), will be on public data, guaranteeing cost-effectiveness and less maintenance concerns. We choose to store and operate with accounts in the system identifying them via an account ID. The mapping between this account IDs and the account private information (such as name, contacts, etc.) should be stored in a in-house private server. Communication with this database need to be encrypted to guarantee compliance with GDPR, write access to it can only be done by passengers while managing their accounts and read writes can be guaranteed only to officials verifying tickets onboard trains. Note that credit card data are stored in the system as tockens, so they are not considered sensitive information.

## Consequences

**Positive Consequences:**

- **Scalability:** Efficient adaptation to variable workloads with serverless, and reliable performance for constant workloads on dedicated servers.

- **Cost-Effectiveness:** Reduced operational costs through serverless computing for dynamic workloads.

- **Performance:** Optimized performance for data-intensive tasks on dedicated servers.

- **Security and Data Privacy:** Enhanced control over sensitive data processing and storage.

**Negative Consequences:**

- **Complexity:** Increased management complexity blending serverless and server environments.

- **Operational Costs:** Potential rise in costs associated with maintaining dedicated server infrastructure.

- **Integration Challenges:** Difficulties in seamless operation between serverless and server-based components.

- **Dependency on Cloud Providers:** Reliance on third-party services for serverless components.

## 2.13 Decision 13: Revenue Division Strategies

### Status

To be decided.

### Architectural Summary

| In the context of | Developing a fair and transparent revenue division strategy among the tycoons within the TrIP system. |
|---|---|
| Facing | The challenge of equitably distributing revenue that accurately reflects each tycoon's contribution to the network. |
| To achieve | A revenue division model that is accepted by all stakeholders and supports the system's long-term sustainability. |
| We considered | Tap Cards Onboard Transport Vehicles, Annual Value-Based Share, Negotiated Shares. |
| And decided for | We advise for Option 1: Tap Cards Onboard Transport Vehicles. |
| Because | It offers a direct link between revenue and service usage, encouraging improvements in service quality and ridership. |
| Accepting | This is fundamentally a business decision that the TrIP board should make. We can accommodate the chosen strategy accordingly. |

### Concern

The main concern of the users in the context of this decision are descibed by the following user stories:

- **User Story 21:** As a tycoon, I want the payment system to accurately track revenue from shared and exclusive segments of my network so that revenue sharing among tycoons is fair and transparent.

- **User Story 23:** As a tycoon, I want the payment system to integrate with my existing infrastructure with minimal disruption so that I can maintain high service levels during the transition.

- **User Story 24:** As a tycoon, I want to access analytics and data insights from the payment system to understand passenger behavior and adjust my services accordingly to maximize revenue and improve service.

### Context

The revenue division strategy impacts the following functional elements and databases within the Train Inter Payment System (TrIP):

- Payment Terminals: Interface for collecting revenue data.

- Tycoon-Specific Systems: Must integrate with revenue distribution logic.

- Databases:

    - Ticket Database: Stores ticket sales data.
    - Payment Database: Tracks completed transactions.
    - Accounts and Subscriptions Database: Manages passenger subscriptions impacting revenue sharing.

- Booking Management Module: Influences revenue calculation based on booked seats.

- Account and Subscription Management Module: Central to managing subscription-related revenue data.

### Criteria

The goal of this decision is to satisfy the user's concerns while safeguarding the other quality attribute requirements of the system. The main QAs affected by this decision are listed here:

- Functional Suitability: Functionally complete, correct, and appropriate method for revenue distribution.

- Performance Efficiency: Strategy must not adversely affect system response times or resource utilization.

- Compatibility: Must coexist and interoperate with the tycoons' diverse systems and databases.

- Security: Ensuring confidentiality and integrity of revenue data.

- Maintainability: Ability to analyze, modify, and test revenue division logic as needed.

- Reliability: Fault tolerance and availability of the revenue division process.

- Flexibility: Adaptability and scalability of the strategy to accommodate new tycoons or changing business models.

## Option 1: Tap Cards Onboard Transport Vehicles

Allocate revenue to the tycoon operating each transport vehicles based on passenger tap card data. This approach directly links revenue to individual train usage.

- *Pros*: Directly correlates revenue with service usage, incentivizing tycoons to improve service quality and increase ridership.

- *Cons*: May not fully account for the value of network-wide contributions, such as infrastructure maintenance or off-peak services.

## Option 2: Annual Value-Based Share

Distribute revenue based on a combination of the value of each tycoon's contributions (infrastructure and services) and ticket sales data from the previous year.

- *Pros*: Acknowledges both the operational and capital contributions of tycoons, potentially offering a more balanced revenue share.

- *Cons*: Risks disadvantaging new entrants or those expanding services, as it relies on historical data.

## Option 3: Negotiated Shares

Tycoons negotiate revenue shares periodically, based on a set of agreed criteria (e.g., service quality, passenger numbers, network investment).

- *Pros*: Allows for flexibility and adaptability in revenue sharing, can directly address the specific contributions and needs of each tycoon.

- *Cons*: Could lead to conflicts or prolonged negotiations, potentially destabilizing the revenue sharing process.

## Decision

While we advise for Option 1: Tap Cards Onboard Transport Vehicles for its direct correlation between revenue and service usage, we recognize that the final decision rests with the TRiP board. This decision is seen as strategic and will significantly impact the partnership dynamic among the tycoons. As such, we emphasize that this choice should be made at the business level, and we are prepared to support and implement the board's decision to best serve the system's objectives and stakeholder needs.

## Consequences

**Positive Consequences:**

- Enhances transparency and fairness in revenue sharing, directly linking earnings to each tycoon's service usage.

- Incentivizes tycoons to focus on improving service quality and expanding ridership.

- Adaptable to system expansions and the introduction of new services or tycoons.

**Negative Consequences:**

- Potential challenges in accurately tracking and attributing revenue, especially in mixed-use or multi-leg journeys.

- May require significant infrastructure investment for full implementation.

- Could lead to disputes over revenue attribution and require continuous oversight and adjustment.

## 2.14   Decision 14: Handling Network Outages

### Status

Open

### Architectural Summary

| In the context of | Ensuring uninterrupted service and data integrity during network outages in the TRiP system. |
|---|---|
| Facing | The challenge of maintaining operational continuity and securing transactions in the absence of network connectivity. |
| To achieve | A resilient system capable of handling ticket validations and transactions securely, even during network disruptions. |
| We considered | Option 1: Delayed Processing; Option 2: Local Caching, |
| And decided for | A hybrid approach utilizing delayed processing and strategic local caching for essential operations. |
| Because | This method balances the need for operational continuity with security and data integrity, catering to the immediate needs of users while addressing potential security risks. |
| Accepting | The complexity of managing dual mechanisms and potential security concerns related to local data storage. |

### Concern

Ensuring system resilience and continuous operation during network disruptions. Related user stories are listed here:

- **User Story 39:** As a passenger, I expect ticket validation to work seamlessly, even during network outages, ensuring that my travel is not hindered by technical issues.

- **User Story 40:** As a tycoon, I require the system to handle all transactions securely and efficiently, irrespective of the current network status, to maintain service continuity and safeguard revenue.

### Context

Network outages pose significant challenges to real-time processing capabilities, affecting ticket validations and financial transactions.

### Criteria

- *Availability*: Minimize service disruption during network outages.

- Ensure data *security* and integrity, especially for financial transactions.

- *Performance*: Optimize local data storage to balance operational continuity with security risks.

### Option 1: Delayed Processing

Transactions are processed once network connectivity is restored, minimizing the need for local data storage but risking transaction backlog.

### Option 2: Local Caching

Local caching of critical data to enable processing during outages, with careful consideration of security implications due to increased local data storage.

### Decision

A combination of Option 1 and strategic caching for essential data will be used for delayed processing. This approach is chosen to maintain service reliability while addressing security concerns. This decision balances the need for uninterrupted service during network outages against the risks associated with local data storage. It considers the priority of user experience and security as per stakeholder requirements and aligns with previous decisions to enhance system resilience.

## Consequences

**Positive Consequences:**

- Maintains ticket validation and system operations during network outages, enhancing user experience.

- Reduces the risk of service disruption, aligning with the expectations of users and tycoons.

- Balances operational needs with security concerns through selective data caching.

**Negative Consequences:**

- May introduce a risk of negative balances due to delayed processing of financial transactions.

- Increases system complexity with the need to manage both delayed processing and data caching mechanisms.

- Poses potential security risks related to the local storage of sensitive data, despite limited caching.

# 3 Context Viewpoint

## 3.1 View: Stakeholders
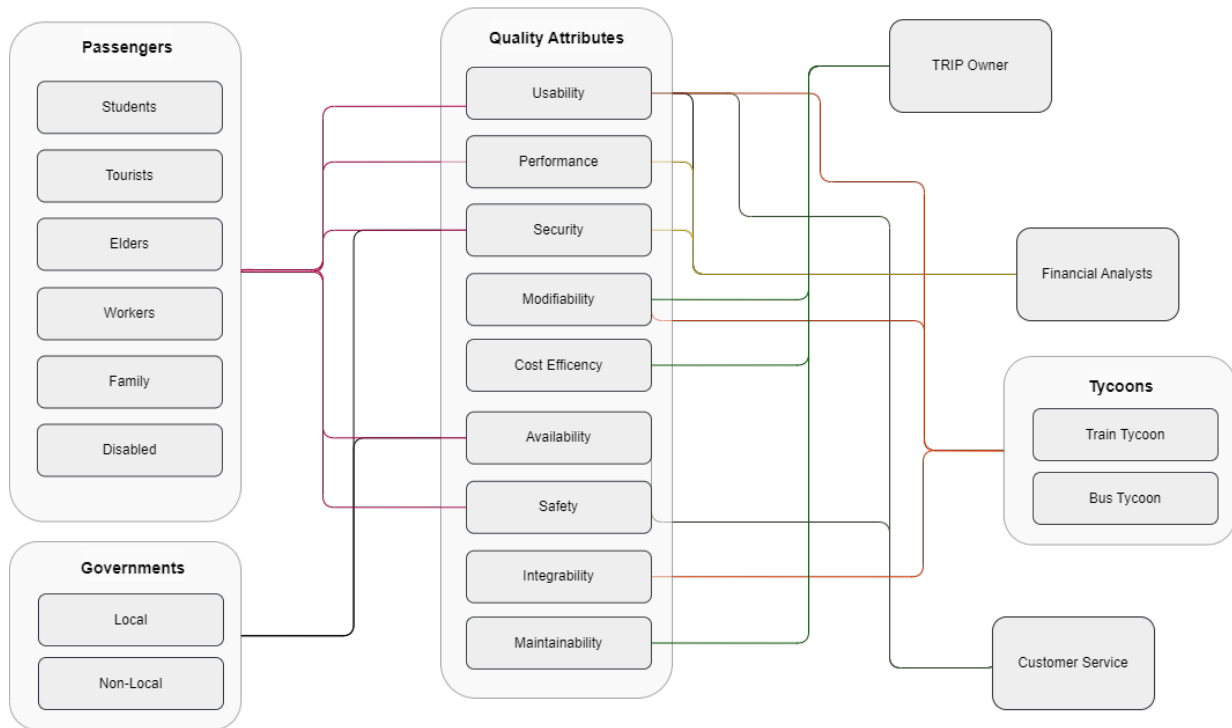
### 3.1.1 Model

**Stakeholder View**



Figure 6: Stakeholder model of the TRIP system.

### 3.1.2 Description

### 3.1.3 Glossary of Elements

### 3.1.4 Analysis on Perspectives

## 3.2 View: Context diagram

### 3.2.1 Model

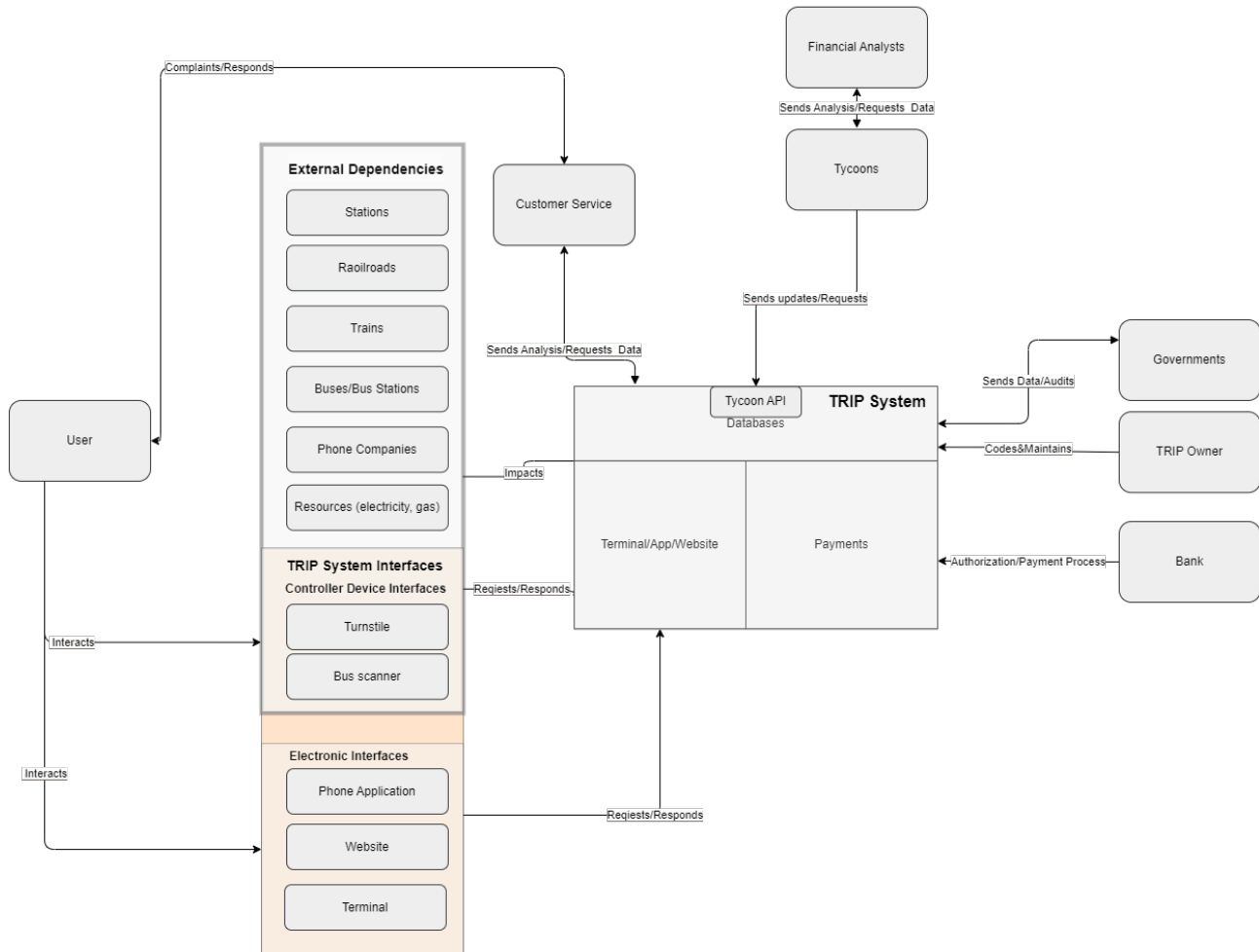**Context View - TRIP System**



Figure 7: Context model of the TrIP system.

### 3.2.2 Description

The Context View of the TRiP SYSTEM delineates the ecosystem within which the system operates, including its interactions with users, external entities, and other system components. At the user level, engagement with the TRiP system is facilitated through various interfaces such as terminals, apps, websites, and controller devices like turnstiles and bus scanners, enabling passengers to access services seamlessly.

The system is subject to a range of external dependencies, including infrastructure elements like stations, railroads, and trains, as well as service providers such as phone companies and utilities that supply essential resources. These components are integral to the system's operations, impacting its functionality and performance.

At the core, the TRiP system is interconnected with Tycoons via the Tycoon API, through which data flows bidirectionally, allowing for the exchange of updates, requests, and analytical data. The databases within the system are pivotal in managing schedules, user accounts, tickets, and payments, all of which are crucial for the day-to-day operations.

The system's architecture is designed to ensure robustness and responsiveness to both the passengers' and Tycoons' needs. It facilitates various processes, from payment transactions, which are securely handled and routed through financial institutions, to the maintenance of service quality, overseen by the TRiP owner and regulated by governmental audits and codes. This comprehensive network of interactions defines the TRiP system's context, emphasizing its multifaceted nature and the critical role it plays in serving its stakeholders.

### 3.2.3  Glossary of the Elements

| Id | Name | Description |
|----|------|-------------|
| 1 | Passenger | Individuals who use the TrIP SYSTEM and its associated services, interacting through various interfaces. |
| 2 | Customer Service | The department that handles passenger complaints and feedback, providing support and sending analysis or data requests to the system. |
| 3 | Financial Analysts | Experts or entities that review financial data, requiring analytical information from the system for decision-making. |
| 4 | Tycoons | The operational decision-makers of the system, possibly managers or algorithms that control system parameters and require data. |
| 5 | Tycoon API | The programming interface through which Tycoons receive updates and send requests to the system. |
| 6 | TrIP System | The core system that integrates various interfaces and processes, forming the central operation platform. |
| 7 | Governments | Regulatory bodies that may require data or perform audits on the system for governance and compliance. |
| 8 | TrIP Owner | The entity or person owning and maintaining the TrIP SYSTEM, responsible for its overall functionality. |
| 9 | Bank | Financial institution that handles the authorization and processing of payments for the system. |
| 10 | Stations | Locations where the TrIP SYSTEM provides service to passengers, such as train or bus stations. |
| 11 | Railroads | Infrastructure providers that offer the tracks on which train services operate. |
| 12 | Trains | The vehicles used by the system to transport passengers from one station to another. |
| 13 | Buses/Bus Stations | The bus services and their stations that are part of the transport network. |
| 14 | Phone Companies | Telecom service providers that facilitate mobile communication and data transfer for the system. |
| 15 | Resources (electricity, gas) | Utility providers that supply essential power and energy required for the system's operations. |
| 16 | Controller Device Interfaces | The interfaces like turnstiles and bus scanners that manage access control and validate user credentials. |
| 17 | Electronic Interfaces | Digital platforms such as mobile applications and websites that passengers interact with for services. |

Table 1: Context model glossary for the TrIP System.

### 3.2.4  Analysis on Perspectives

## 4   Functional Viewpoint

We start discussing the Functional Viewpoint by analyzing stakeholder concerns that need to be addressed. We individuated the following user stories:

- User stories TODO

As a consequence, we decided to prioritize Quality Attributes for this view as indicated in Table 2.

| | Usability | Performance | Security | Modifiability | Cost Efficiency | Availability | Safety | Integrability | Maintainability |
|---|---|---|---|---|---|---|---|---|---|
| Functional View | X | X | | | X | X | | | X |

Table 2: Functional View Prioritized Quality Attributes

## 4.1 View: Main Functional Elements

### 4.1.1 Model

**Functional view - TRIP System**
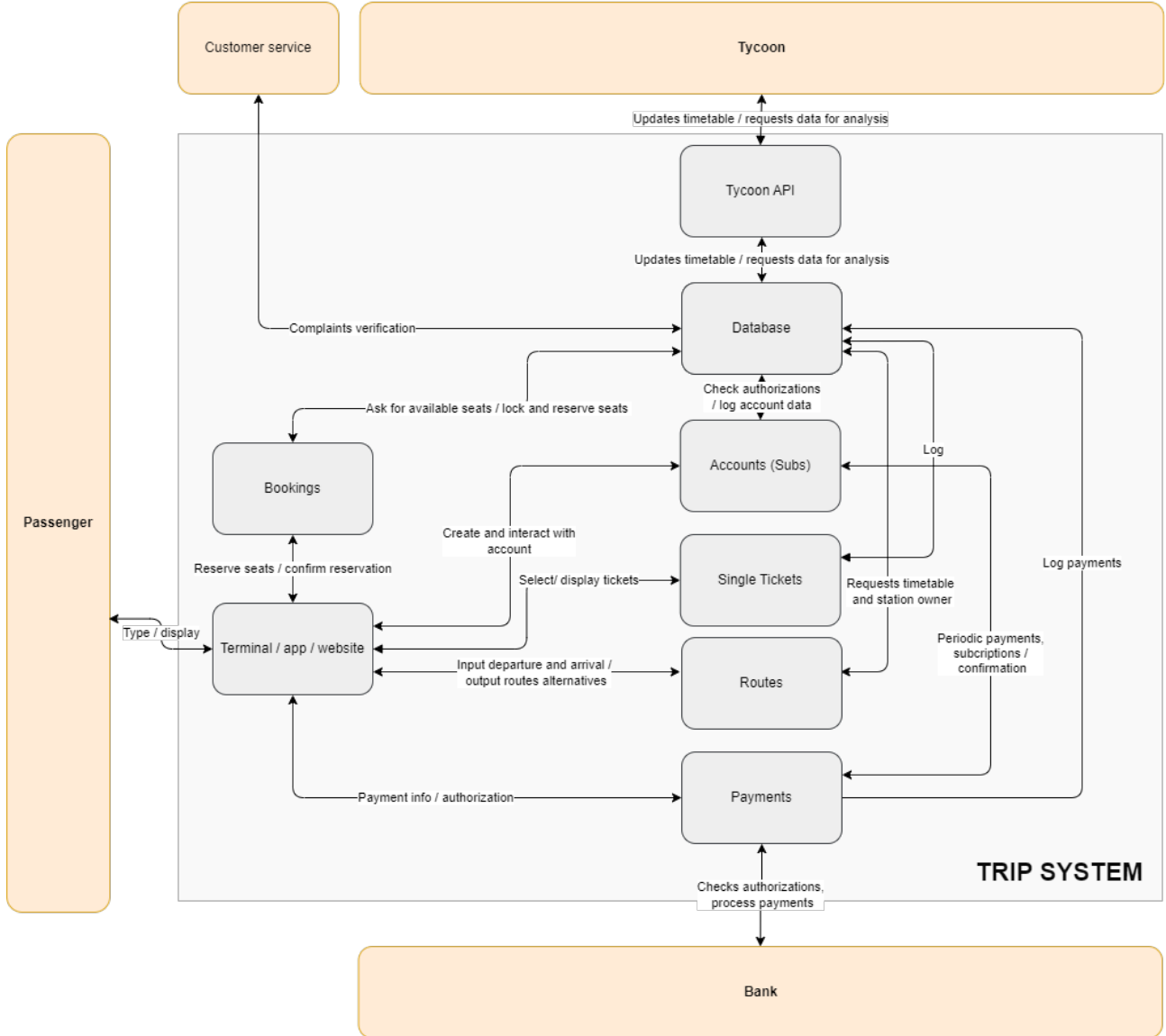


Figure 8: TrIP System.

### 4.1.2 Description

The functional view diagram of the TrIP system illustrates the major functions of the system and how they interact with each other. The diagram features boxes representing different functions, connected by arrows that represent interactions. The system starts with the user, who can interact with the system through various interfaces, such as terminals, apps, or websites. The user can input their departure and arrival stations, and the system will then display a list of available routes. The user can then select a route and proceed to payment. The payment system can handle various payment methods, including single tickets, subscriptions, and fillable cards. If the user has a subscription, the system will automatically check if the subscription is valid for the

selected route. This is done by the Account and Subscription Management module, which communicates with the tycoon systems to verify the subscription. If the subscription is not valid, the user will be prompted to purchase a single ticket or top up their fillable card. Once the payment is processed, the system will generate a ticket or update the user's travel card. The user can then scan their ticket or card at the turnstile to gain access to the train platform. The turnstile communicates with the Account and Subscription Management module to verify the ticket or card and update the passenger's state. The system also includes a number of other functions, such as a booking system, a route optimization system, and a customer service system. The booking system allows users to reserve seats on trains. The route optimization system helps users find the most efficient routes between their departure and arrival stations. The customer service system provides support to passengers with inquiries and complaints. The system is designed to be scalable and flexible, so that it can be easily adapted to accommodate new tycoons and changing business models. The system is also designed to be secure, so that passenger data is protected from unauthorized access.

### 4.1.3 Glossary of Elements

| Id | Name | Description |
|---|---|---|
| 1 | Passenger | End-users of the TrIP SYSTEM who interact with various system components to manage their travel experience. |
| 2 | Customer Service | The interface for passengers to make inquiries or complaints and receive assistance with bookings or account issues. |
| 3 | Tycoon | The administrative or business logic module that updates timetables and analyzes system data for improvements or reporting. |
| 4 | Database | An abstraction for the set of databases that stores all system data including passenger accounts, bookings, and payment information. Detailed information about how different databases are handled is detailed in the Information View. |
| 5 | Bookings | The system component where passengers can inquire about seat availability and make reservations. |
| 6 | Accounts (Subs) | The system managing passenger accounts and subscriptions, responsible for authorization checks and account data logging. It is is also responsible for single tickets and fillable cards, as they can be seen as temporary and anonymous accounts. |
| 7 | Routes | The component that manages route information and provides passengers with timetables, station ownership details, and route alternatives. |
| 8 | Payments | The module handling all financial transactions, including passenger payments and periodic billing. |
| 9 | Bank | The financial institution interface for authorizing and processing payments linked to the system. |
| 10 | Terminal/App/Website | User interfaces through which they can access services such as booking, route information, and payment. |

Table 3: Glossary of elements detailing the components of the TrIP SYSTEM and their roles in facilitating user interaction and service provision.

### 4.1.4 Analysis on Perspectives

## 4.2 View: Ticket scanning

### 4.2.1 Model

**Functional View - Scanning Procedure**
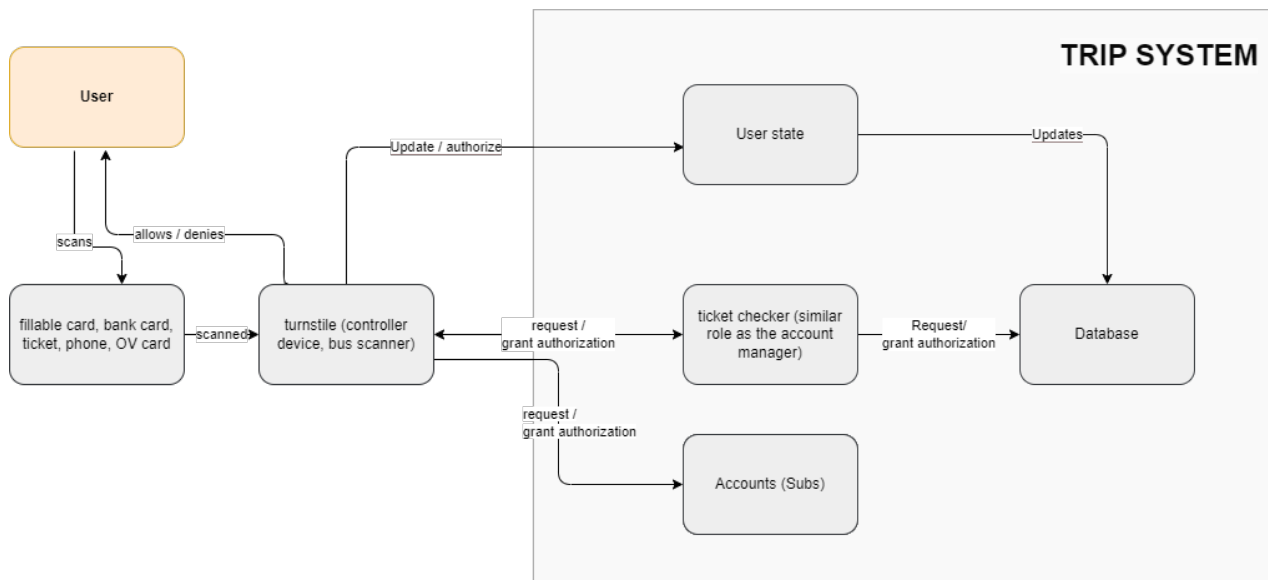


Figure 9: Interaction with a ticket scanner.

### 4.2.2 Description

The scanning procedure within the TRIP SYSTEM encapsulates the interactions between the passenger and the system's access control mechanisms. It begins with the user presenting a valid form of transit access—such as a card or mobile device—to a scanning device like a turnstile or bus scanner. This device then consults the User state, a repository of the passenger's authorization status, to allow or deny entry. In parallel, the Ticket Checker function verifies the user's credentials against the Accounts subsystem, which manages detailed account information and subscriptions. Any changes to the user's status are updated in real time in the central Database, ensuring accurate tracking of access and travel history. This process ensures a secure, streamlined experience for passengers while providing the system with the necessary oversight to prevent unauthorized access

#### 4.2.3 Glossary of Elements

| Id | Name | Description |
|---|---|---|
| 1 | Passenger | The individual who uses the trip system and interacts with various components such as turnstiles and ticket checkers. |
| 2 | Fillable Card, Bank Card, Ticket, Phone, OV Card | Various forms of identification or payment methods that the passenger can use within the system. These are scanned by the turnstile to allow or deny access. |
| 3 | Turnstile (Controller Device, Bus Scanner) | A physical barrier or scanner that reads the passenger's ticket or card and determines whether to grant or deny access based on the passenger state or account information. |
| 4 | Passenger State | A system component that maintains the current state of the passenger within the system, including authorization and access rights, which is updated upon passenger interaction with the turnstile. |
| 5 | Ticket Checker (Account Manager) | An agent or system role similar to the account manager that requests or grants authorization for passenger access, potentially by checking the passenger state against the database. |
| 6 | Accounts (Subs) | The subsystem managing passenger accounts and subscriptions, which may interact with the turnstile and ticket checker to verify and update passenger access rights. |
| 7 | Database | An abstraction for the set of databases that stores all system data including passenger accounts, bookings, and payment information. Detailed information about how different databases are handled is detailed in the Information View. |

Table 4: Glossary of elements for the Functional View - Turnstiles, detailing the components and their roles in passenger access and authorization within the TrIP SYSTEM.

#### 4.2.4 Analysis on Perspectives

| Aspect | Details |
|---|---|
| Source | Passenger interacts with the payment terminal UI. |
| Stimulus | The passenger navigates the options to buy a ticket. |
| Artifact | Standardized User Interface (UI) of the payment terminal. |
| Response | The UI displays a clear, consistent, and intuitive navigation path for ticket purchase. |
| Measure | 95% of passengers successfully purchase tickets without assistance. |

Table 5: Scenario for Usability & Cost Efficiency - Standardized UIs

| Aspect | Details |
|---|---|
| Source | A new tycoon's system attempting to integrate with the TrIP system. |
| Stimulus | The tycoon sends a request to access route and fare data. |
| Artifact | Tycoon API that standardizes data exchange with the TrIP system. |
| Response | The API facilitates the integration, providing access to the required data. |
| Measure | Integration is completed within 3 business days, with zero errors in data format conversion. |

Table 6: Scenario for Scalability via Tycoon API

| Aspect | Details |
|---|---|
| Source | Passengers with varying preferences for payment, each approaching the transit system's access points. |
| Stimulus | Passengers select their payment method of choice, ranging from physical cash for single rides to digitally managed subscriptions, and some opt for the convenience of pre-loaded fare cards. |
| Artifact | Integrated payment and authentication platform within the TrIP system. |
| Response | The system adeptly manages the assortment of transactions, crediting cash payments, verifying subscription validity, and debiting pre-loaded cards, all in a streamlined fashion. |
| Measure | The system consistently processes transactions of all types with a rapid response rate, registering a less than 2-second average processing time and maintaining a transaction success rate of 99%. |

Table 7: Scenario for Usability via Accommodating Varied Payment Preferences

**Scenarios**   In addressing the Quality Attribute (QA) priorities highlighted by users, our functional viewpoint incorporates several key decisions designed to enhance usability, maintainability, scalability, and performance efficiency.

To meet the usability needs of users, we have opted for standardized User Interfaces (UIs). These UIs, being open-source and widely utilized, benefit from a large user base that contributes to their maintenance and robustness. This choice ensures a user-friendly and reliable interface for passengers over the long term. Furthermore, it aligns with the TrIP owner's preferences by offering ease of maintenance and low operational costs.

The introduction of multiple payment methods, including fillable cards, credit cards, and single tickets, significantly simplifies passenger interaction with the TrIP system. The integration of accounts and subscriptions further facilitates seamless travel across multiple tycoon networks, enabling passengers to efficiently manage and use their subscriptions.

In response to Event 1, which underscored the need for easier integration of new tycoons, we have implemented the Tycoon API. This API standardizes data requests from each tycoon, irrespective of their mode of transportation, thus facilitating the smooth integration of new tycoons into the system. The Tycoon API, in conjunction with the accounts and bookings module, ensures that passengers can effortlessly utilize their subscriptions across different networks and book their preferred routes.

Event 3, which focused on traffic jams, raised the importance of optimizing system performance during peak periods. By deciding on a centralized route management module, we have streamlined data flow between the TrIP system and the tycoons. This module not only simplifies data management but also enhances the system's ability to handle high request volumes through optimization and caching strategies.

In summary, the functional viewpoint of our system meticulously addresses the initial QA priorities, along with the challenges presented by new tycoon integration (Event 1) and traffic jams (Event 3). These decisions collectively ensure a user-friendly, scalable, and high-performing TrIP system for passengers, tycoons, and the TrIP owner alike.

# 5   Information Viewpoint

We start discussing the Information Viewpoint by analyzing stakeholder concerns that need to be addressed. We individuated the following user stories:

- User stories TODO

As a consequence, we decided to prioritize Quality Attributes for this view as indicated in Table 8.

| | Usability | Performance | Security | Modifiability | Cost Efficiency | Availability | Safety | Integrability | Maintainability |
|---|---|---|---|---|---|---|---|---|---|
| Information View | | | X | | | | | | |

Table 8: Information View Prioritized Quality Attributes

## 5.1 View: Main Information View
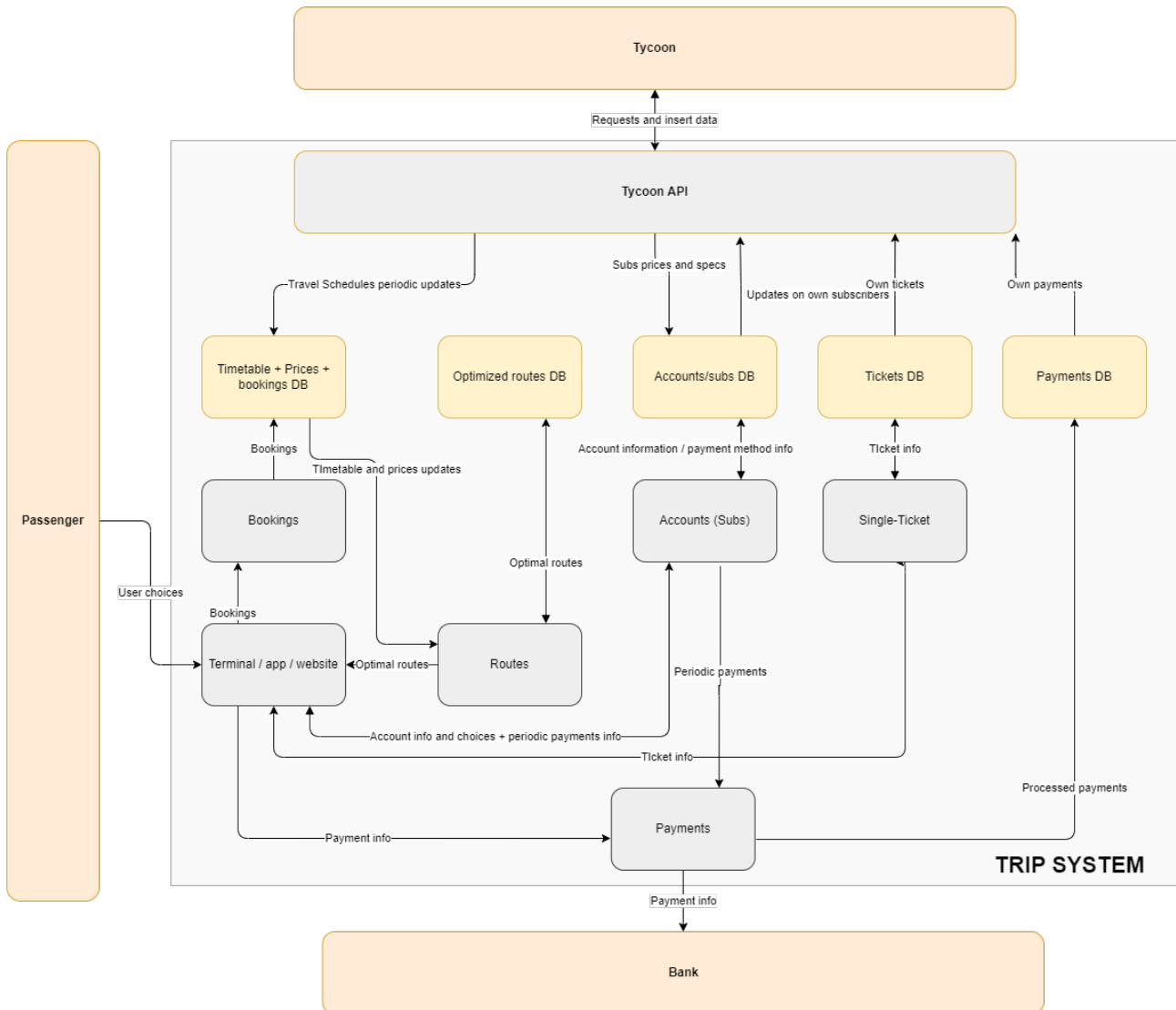
### 5.1.1 Model

**Information View**



Figure 10: Information view.

### 5.1.2 Description

The Information View of the TrIP SYSTEM is about how data moves and is stored. The system uses several databases. The Timetable + Prices + Bookings DB keeps track of schedules, prices, and reservations. The Optimized Routes DB has data on the best travel paths. The Accounts/Subs DB holds information on user accounts and subscriptions. The Tickets DB records ticket purchases, and the Payments DB keeps a record of all payments.

Tycoons use the Tycoon API to put in and get data. Passengers use terminals, apps, or websites to book travel and select routes. This choice and payment info go into the databases, updating the system.

The Routes component uses the Optimized Routes DB to give passengers the best travel options. The Accounts (Subs) part handles subscription details and payments, working with the Accounts/Subs DB. For single tickets, there's a separate interface that works with the Tickets DB. Payments process transactions and send completed payment information to the bank.

### 5.1.3 Glossary of Elements

Table 9: Glossary of elements for the Information View of the TrIP SYSTEM.

| Element | Description |
|---------|-------------|
| Tycoon | Tycoons responsible for making requests for analysis, and inserting travel data. |
| Tycoon API | Tycoon's way of interacting to the TrIP system. |
| Timetable + Prices + bookings DB | Stores data regarding travel schedules, pricing, and booking information. |
| Optimized routes DB | Contains information on the most efficient travel routes that have been calculated and stored. |
| Accounts/Subs DB | Maintains records of user accounts and their subscriptions for travel services. |
| Tickets DB | A database that logs ticket purchases and holds ticket-related information. |
| Payments DB | Records and processes transactions related to payments within the system. |
| Passenger | The end-user or customer who utilizes the system for travel services. |
| Bookings | The system component or interface that passengers interact with to manage and view their bookings. |
| Terminal / App / Website | The various platforms through which passengers can access the system for services. |
| Routes | Involves the determination and selection of travel routes within the system. |
| Accounts (Subs) | Manages the subscription details associated with user accounts. |
| Single-Ticket | A system or interface that deals with the purchase of individual travel tickets. |
| Periodic payments | Manages recurring payments, typically for subscription services. |
| Payments | Processes transactions related to immediate payments within the system. |
| Processed payments | A log or record of payment transactions that have been completed. |
| Bank | The financial institution where final payment transactions are processed and funds are transferred. |

### 5.1.4 Analysis on Perspectives

## 5.2 View: Scanning information flows

### 5.2.1 Model

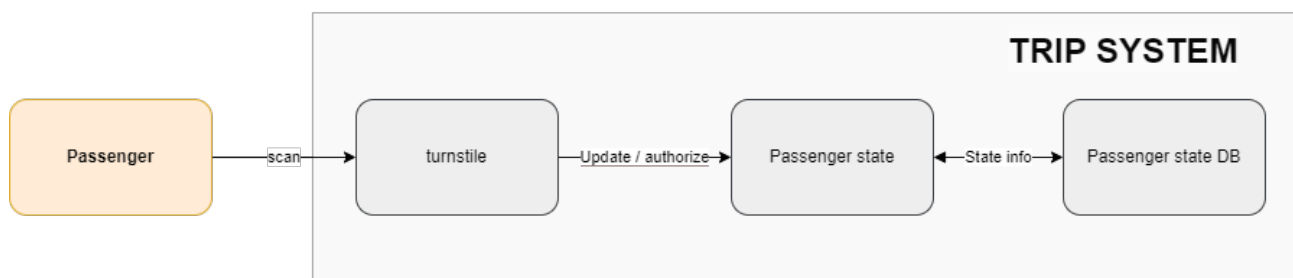## Information View - Scanning Procedure



Figure 11: Information view related to the scanning procedure.

### 5.2.2 Description

The TrIP SYSTEM's scanning procedure is a sequence that starts with the Passenger, who scans at the Turnstile to gain entry. This triggers the Update/Authorize process, updating the Passenger State with the user's access

rights. The Passenger State reflects the user's current permissions within the TRIP SYSTEM. Each user's information is recorded in the Passenger State DB, a database that tracks and manages user statuses.

Table 10: Legend for the Scanning Procedure in the Information View of the TrIP SYSTEM.

| Id | Name | Description |
|---|---|---|
| 1 | Passenger | The starting point representing the individual using the TrIP SYSTEM. |
| 2 | Turnstile | The physical or virtual entry point where the passenger scans to gain access. |
| 3 | Update / Authorize | The process that updates the system and authorizes the passenger to proceed. |
| 4 | Passenger State | The current status of the passenger within the system, which is updated after scanning. |
| 5 | Passenger State DB | The database that records the state information of the passenger. |

### 5.2.3 Analysis on Perspectives

| Aspect | Details |
|---|---|
| Source | Passenger utilizing the TrIP system to manage their travel plans. |
| Stimulus | The passenger enters personal and travel data into the system. |
| Artifact | Encrypted Account Lookup Database within the TrIP system. |
| Response | The system securely stores the passenger's data using state-of-the-art encryption both at rest and in transit, ensuring that personal and travel information is not accessible by unauthorized entities. Access to this data is restricted to specific operations within the TrIP system that require passenger verification. |
| Measure | Passenger data breaches are non-existent, and passengers report a high level of trust in the system's data protection capabilities. Security logs show no unauthorized access, affirming the integrity of the encryption measures. |

Table 11: Scenario for Security - Passenger Data Protection

| Aspect | Details |
|---|---|
| Source | Passenger making a payment through the TrIP system. |
| Stimulus | The passenger chooses a payment method and initiates a transaction. |
| Artifact | Bank interfaced with TrIP system for processing payments. |
| Response | The bank processes the payment with rigorous security protocols, ensuring that the payment information is handled in a secure, encrypted channel. The system guarantees that payment data is never stored within the TrIP system and is managed exclusively by the bank, providing an additional layer of security. |
| Measure | Payment transactions are completed with zero incidents of data compromise, and the response time for transaction completion remains within 3 seconds. Post-transaction audits confirm the absence of payment data retention within the TrIP system. |

Table 12: Scenario for Security - Secure Payment Transactions

| Aspect | Details |
|---|---|
| Source | Passenger subscribing to a specific tycoon's services within the TrIP system. |
| Stimulus | The subscribed tycoon requests access to the passenger's account data for service personalization. |
| Artifact | Tycoon API enabling access control to passenger data within the TrIP system. |
| Response | The TrIP system authenticates the tycoon's request through the Tycoon API, which validates that only the tycoon with whom the passenger is subscribed can retrieve the required data. This ensures that each tycoon can access only their passengers' data, thus maintaining strict data access control and privacy. |
| Measure | Regular audits indicate 100% compliance with data access policies, with no reported incidents of data leakage or unauthorized access. Passenger feedback confirms trust in the data privacy practices of the TrIP system. |

Table 13: Scenario for Security - Controlled Tycoon Access to Passenger Data

**Scenarios**  The Information View of the TrIP system provides a comprehensive architecture designed to handle and safeguard data movement and storage, ensuring the system's integrity and security.

The design segregates data across specialized databases, including those for Timetables + Prices + Bookings, Optimized Routes, Accounts/Subs, Tickets, and Payments. This segregation follows best practices in database security, preventing any single point of compromise and ensuring each database only holds data relevant to its function. Access to these databases is strictly regulated, with the system enforcing robust authentication and authorization mechanisms to verify and control interactions.

Encryption plays an important role in the system, particularly within the Accounts/Subs Database. It protects sensitive passenger information by using encrypted account lookup database. Data at rest within this database, as well as data in transit to and from it, is encrypted, thereby securing personal and travel information from unauthorized access.

Payments are a critical interaction point for the system and its users. The TrIP system approaches this by interfacing directly with banks for payment processing. This interface means that the sensitive financial information is managed within the secure domain of banking institutions, which are inherently designed to handle such data securely. This direct interfacing also implies that the TrIP system does not store payment details, which greatly reduces its liability and exposure to financial data breaches.

The Tycoon API serves as a broker for data access by tycoons. It provides a controlled access point, permitting only authorized tycoons to retrieve passenger data. This access control mechanism is crucial for maintaining passenger privacy and data security. It allows tycoons to access only the data necessary for their service offerings and nothing beyond that.

Using the scenarios provided, we can see how the system's design and technology choices map directly to its operations. Passengers entering their information into terminals or apps can be assured that their data will be encrypted and stored securely, as detailed in the first scenario. Payment processes, as per the second scenario, happen within the secure confines of the banking system. Lastly, the way the Tycoon API manages data requests ensures that passenger data is only accessed by authorized tycoons, enhancing privacy and data protection as outlined in the third scenario.

# 6  Concurrency Viewpoint

We start discussing the information viewpoint by analyzing stakeholder concerns that need to be addressed. We individuated the following user stories:

- User stories TODO

As a consequence, we decided to prioritize Quality Attributes for this view as indicated in Table 14.

| | Usability | Performance | Security | Modifiability | Cost Efficiency | Availability | Safety | Integrability | Maintainability |
|---|---|---|---|---|---|---|---|---|---|
| Concurrency View | | X | | | | X | X | | |

Table 14: Concurrency View Prioritized Quality Attributes

## 6.1 View: Ticket purchase

### 6.1.1 Model

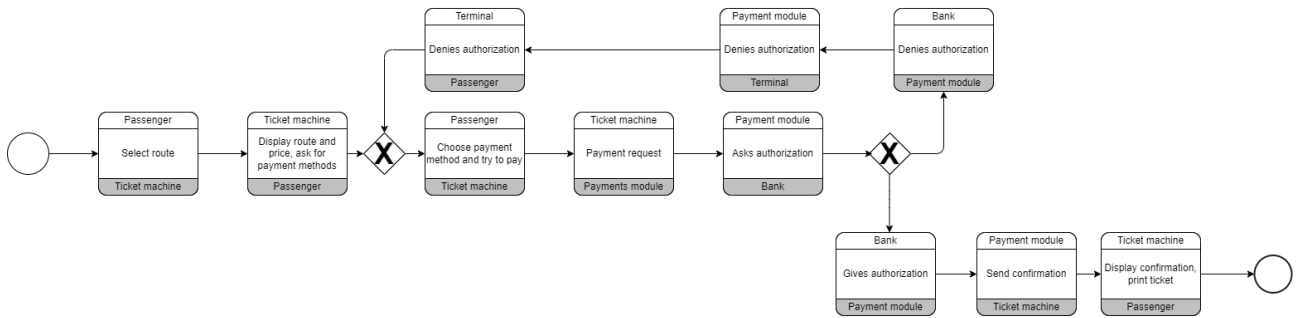**Concurrency View - Ticket purchase**



Figure 12: Concurrency view related to ticket purchase.

### 6.1.2 Description

In the TRIP SYSTEM, the Passenger starts the process at the Ticket Machine, choosing a route and payment method. The Terminal takes over to process the payment, and the Payment Module communicates with the Bank for payment approval. If the Bank approves, the Payment Module signals the Ticket Machine to confirm the transaction and print the ticket. If the Bank denies the payment, the process is stopped, and the Passenger is notified.

Table 15: Glossary for the Payment and Ticketing Process.

| Element | Description |
|---|---|
| Passenger | The customer who initiates the process by selecting a route and choosing a payment method to pay for a ticket. |
| Ticket Machine | The interface through which the passenger selects a route, displays the route and price, and chooses a payment method. |
| Terminal | The point of service where the passenger's payment authorization is processed. |
| Payment Module | The system component that interacts with the bank to request payment authorization for the transaction. |
| Bank | The financial institution that either authorizes or denies the payment transaction. |

### 6.1.3 Analysis on Perspectives

## 6.2 View: Scanning procedure

### 6.2.1 Model
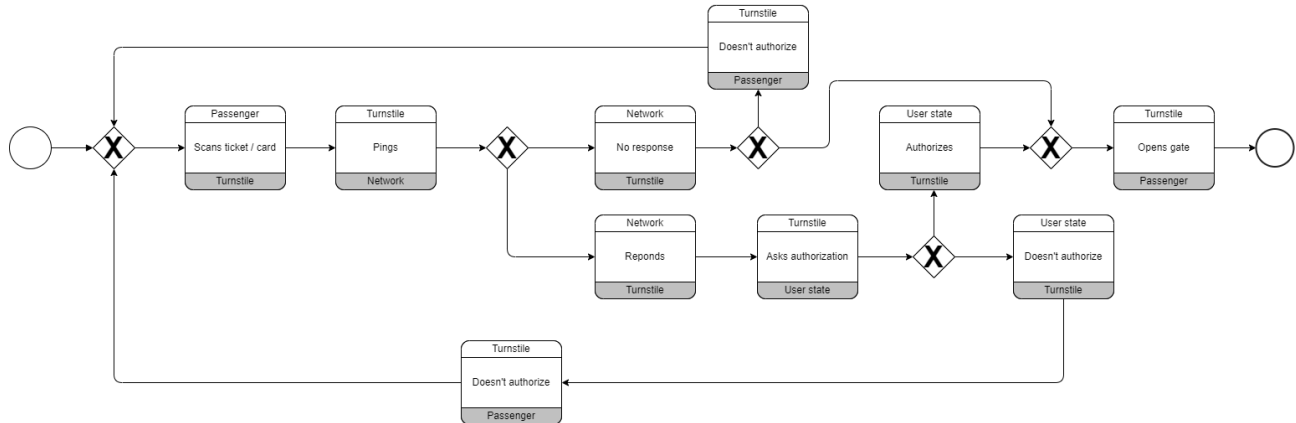
**Concurrency View - Scanning Procedure**



Figure 13: Concurrency view related to the scanning procedure.

### 6.2.2 Description

The diagram depicts a turnstile access procedure with conditional paths based on network availability and user authorization. When a Passenger scans their ticket or card at the Turnstile, two scenarios can unfold:

1. If the Turnstile is unable to connect to the Network (no response), it defaults to a local decision. In this case, the turnstile can still potentially authorize the passenger to proceed based on offline data or preconfigured rules.

2. If the Turnstile connects to the Network (network responds), it then requests authorization from the User State. The User State, after evaluating the request, either grants or denies authorization:

   - If the User State authorizes the request, the Turnstile receives a signal to open the gate, allowing the Passenger to pass.
   - If the User State denies the request, the Turnstile will not open, and the Passenger is not allowed to proceed.

This process ensures that the system can function and provide decisions autonomously, even in the absence of network connectivity, enhancing reliability and user experience.

### 6.2.3 Glossary of Elements

Table 16: Glossary for Turnstile Interaction Process.

| Component | Description |
| --- | --- |
| Passenger | An individual who is attempting to gain entry through the turnstile by scanning a ticket or card. |
| Turnstile | A physical barrier at an entry point that controls access, typically based on ticket or card validation. |
| Network | The communication system that the turnstile interfaces with to verify access rights. |
| User State | A system component that maintains the current state of a user's access rights, determining whether entry is authorized. |

### 6.2.4 Analysis on Perspectives

## 6.3 View: Tycoon data updates

### 6.3.1 Model
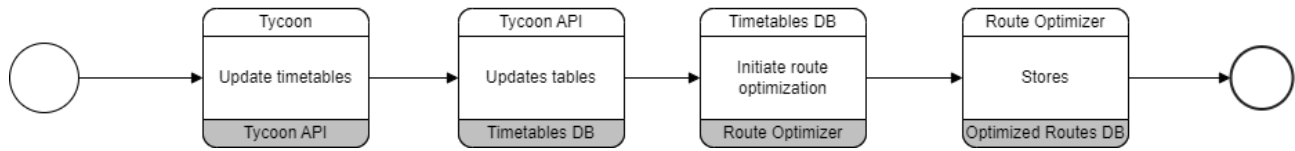
## Concurrency View - Tycoon Data Update



Figure 14: Concurrency view related to tycoons data updates.

### 6.3.2 Description

This process outlines how timetable updates and route optimization are handled in the system. It begins with the Tycoon, an administrative role, who updates timetables through the Tycoon API. These updates are then applied to the Timetables DB. Following this update, the Route Optimizer initiates the optimization process, which takes the updated timetable data to determine the most efficient routes. These optimized routes are then stored in the Optimized Routes DB, completing the cycle of updating and optimizing the route information available to the system and its users.

Table 17: Glossary for Route Optimization Process.

| Component | Description |
|---|---|
| Tycoon | Represents the train companies or transport entities responsible for managing train schedules and routes. |
| Tycoon API | The application programming interface that allows the tycoons to update and access timetable information. |
| Timetables DB | The database where train schedules, routes, and associated data are stored and updated. |
| Route Optimizer | The system component that calculates the most efficient routes, likely using algorithms to process timetable data. |
| Optimized Routes DB | A specialized database that stores the results of the route optimization process. |

### 6.3.3 Glossary of Elements

### 6.3.4 Analysis on Perspectives

| Aspect | Details |
|---|---|
| Source | Network outage impacting the TrIP system's connectivity. |
| Stimulus | A passenger attempts to validate their ticket at a turnstile during the outage. |
| Artifact | Hybrid approach of delayed processing and local caching implemented in the TrIP system. |
| Response | The turnstile, equipped with local caching, validates the passenger's ticket against stored data, allowing entry. Transaction details are queued for delayed processing once network connectivity resumes, ensuring no service disruption. |
| Measure | Passenger service continuity is maintained with 99.9% uptime, and ticket validations during outages show no significant delay, ensuring high system availability. |

Table 18: Scenario for Availability - Network Outage Handling

| Aspect | Details |
| --- | --- |
| Source | TrIP system's safety protocols for access control. |
| Stimulus | An individual attempts to access the transit system without a valid ticket or permission. |
| Artifact | Turnstile with integrated validation system that checks for valid tickets. |
| Response | The turnstile denies entry to the individual without a valid ticket, preventing unauthorized access and maintaining the safety and security of the transit environment. |
| Measure | Zero reported incidents of unauthorized access, ensuring the safety of the transit system and its users. |

Table 19: Scenario for Safety - Authorized Access Enforcement

| Aspect | Details |
| --- | --- |
| Source | Passenger requests the best route options via the TrIP system during peak hours. |
| Stimulus | High demand for route information causes potential strain on the system. |
| Artifact | Optimized Routes Database that stores frequently requested route data. |
| Response | The TrIP system quickly retrieves optimized route options from the pre-stored data, providing the passenger with fast and efficient service without needing to compute the routes in real-time. |
| Measure | The response time for route information requests during peak periods remains under 2 seconds, enhancing the overall performance of the system. |

Table 20: Scenario for Performance - Efficient Route Retrieval

**Scenarios** The Concurrency Viewpoint is central to the TRiP system's architecture, focusing on ensuring that user interactions with the system occur smoothly and without delay, even when multiple operations are executed simultaneously. This viewpoint addresses the system's capacity to manage concurrent actions effectively, with a particular emphasis on maintaining performance, safety, and availability.

**Ticket Purchasing Process:**
The sequence begins with the passenger selecting a route and payment method via the Ticket Machine. This process is crucial for performance, as it demands a fast and seamless operation to prevent queues and delays. Once the payment method is chosen, the Terminal communicates with the Payment Module to process the transaction. The Payment Module's role here is two-fold: to ensure the safety of the transaction by validating payment with the bank and to maintain the system's performance by providing quick feedback to the Terminal. If the bank authorizes the payment, the Payment Module prompts the Ticket Machine to dispense a ticket, showcasing the system's availability to complete transactions without delay. In case of a denial, the passenger is promptly informed, allowing them to take alternative actions, which is vital for maintaining system usability and ensuring the passenger's experience remains positive.

**Turnstile Access Control:**
The second aspect of concurrency comes into play at the turnstiles, where passengers scan their tickets or cards to gain access to the transport services. In this operation, the system's availability and safety are the primary concerns. The Turnstile must quickly authorize entry to avoid creating bottlenecks, which it does by pinging the Network to retrieve the User State information. If the Network is unresponsive, the Turnstile uses locally cached data to make an authorization decision, ensuring continuous operation and thus maintaining high availability. When the Network responds, the User State determines access permission. This dual approach ensures that access control is robust against network issues, preserving the integrity of operations and passenger safety by preventing unauthorized access.

**Route Optimization Updates:**
Finally, the system's capacity to update and optimize routes through the Tycoon API and Route Optimizer reflects its modifiability and performance attributes. Tycoons can update timetables, which are then processed to optimize routes, and these optimizations are stored in the Optimized Routes DB for quick retrieval. This ensures that passengers receive up-to-date and efficient routing information, which is especially important during peak traffic conditions.

By detailing these processes, the concurrency viewpoint showcases the TrIP system's capability to maintain a high level of service and security across its operations, aligning with the defined quality attributes of performance, safety, and availability.

# 7 Deployment Viewpoint

We start discussing the information viewpoint by analyzing stakeholder concerns that need to be addressed. We individuated the following user stories:

- User stories TODO

As a consequence, we decided to prioritize Quality Attributes for this view as indicated in Table 21.

| | Usability | Performance | Security | Modifiability | Cost Efficiency | Availability | Safety | Integrability | Maintainability |
|---|---|---|---|---|---|---|---|---|---|
| Deployment View | | X | X | | | | | | |

Table 21: Deployment View Prioritized Quality Attributes

## 7.1 View: Scanning procedure
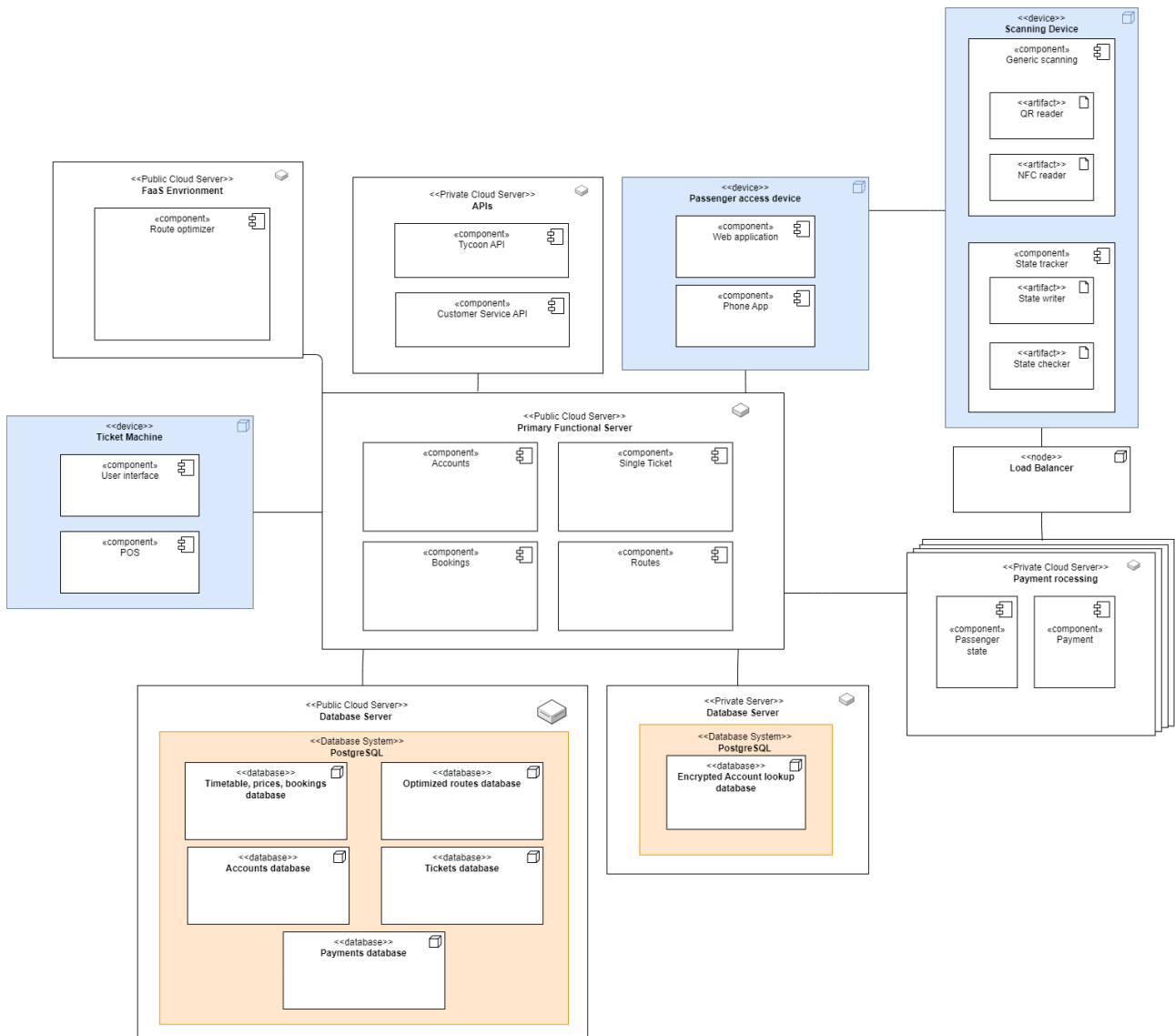
### 7.1.1 Model



Figure 15: Deployment view related to the scanning procedure.

### 7.1.2 Description

The deployment view diagram illustrates the TrIP system's architecture, showing components distributed across public and private cloud servers. Public cloud servers house the FaaS Environment for route optimization, a Terminal with Customer Service API, and the Primary Functional Server hosting Accounts, Single Ticket, Bookings, and Routes components. Devices for passenger access, such as web and phone apps, interface with scanning devices like QR and NFC readers, managed by state trackers and checkers.

The ticket machine interfaces include User Interface and POS components. Database servers in the public cloud manage the PostgreSQL databases for Timetables, Prices, Bookings, Optimized Routes, Accounts, Tickets, and Payments. The private cloud server secures the Encrypted Account Lookup database. A Load Balancer ensures efficient traffic management.

Payment processing, crucial for managing Passenger State and payments, is securely handled in the private cloud, emphasizing the system's focus on security and reliable transaction handling.

### 7.1.3 Glossary of Elements

Table 22: Glossary for the Deployment View of the TrIP SYSTEM.

| Component | Description | Hosted On |
|---|---|---|
| FaaS Environment | Provides a platform for executing backend functions in a serverless architecture, such as route optimization. | Public Cloud Servers |
| Tycoon API | Interfaces allowing tycoons' software and systems to communicate with the TrIP system. | Private Cloud Servers |
| Customer Service API | Facilitates customer service operations by providing data access and manipulation capabilities. | Private Cloud Servers |
| Passenger Access Device | Equipment that passengers interact with to access the system, like ticket validation and purchases. | Terminal, App, Website |
| Ticket Machine | Physical machines where passengers can purchase tickets and manage their bookings. | Station Locations |
| Scanning Device | Tools used for reading ticket information, necessary for entry validation and passenger tracking. | Entry/Exit Gates |
| QR Reader | Specialized scanners that interpret QR codes on tickets for entry validation or information retrieval. | Scanning Devices |
| NFC Reader | Contactless devices that read NFC tags for authentication and ticket validation. | Passenger Access Devices |
| Timetable, Prices, Bookings Database | Stores all relevant data for scheduling, pricing, and passenger bookings. | Public Cloud Servers |
| Optimized Routes Database | Contains data on the most efficient routes calculated by the optimization algorithms. | Public Cloud Servers |
| Accounts Database | Manages passenger account information, including subscription details and personal data. | Public Cloud Servers |
| Tickets Database | Holds data related to ticket sales, validations, and historical transactions. | Public Cloud Servers |
| Payments Database | Processes and records all payment transactions within the system. | Public Cloud Servers |
| Encrypted Account Lookup Database | A secure database that stores sensitive account information, accessible only through authorized queries. | Private Cloud Servers |

### 7.1.4 Analysis on Perspectives

| Aspect | Details |
|---|---|
| Source | TrIP system experiencing a surge in demand due to a new university opening at a station. |
| Stimulus | Passenger numbers have more than quadrupled, causing performance delays during rush hours. |
| Artifact | The deployment of the TrIP system across public cloud servers and private cloud servers. |
| Response | The system's cloud infrastructure is updated to leverage elastic scaling capabilities. Load balancers distribute the increased traffic efficiently, and additional computing resources are allocated to handle the high load, particularly for route optimization and payment processing components. |
| Measure | Post-update, the TrIP system handles a fourfold increase in user load with response times reduced by 70%, ensuring that all passengers complete their transactions and receive route information within 2 seconds, even during peak rush hours. |

Table 23: Scenario for Performance - Cloud Infrastructure Scaling

| Aspect | Details |
|---|---|
| Source | TrIP system architecture analysis in response to a competitor's security breach. |
| Stimulus | The need to verify and bolster the security measures in place to protect passenger data. |
| Artifact | Deployment architecture of the TrIP system with a focus on data servers and encryption protocols. |
| Response | The system utilizes advanced encryption for data at rest and in transit within private cloud servers. Access controls are strictly enforced, ensuring only authenticated and authorized components interact with sensitive data. Periodic security audits and real-time intrusion detection systems are in place to monitor and immediately respond to any unauthorized access attempts. |
| Measure | Security tests confirm no data leakage, and the system's compliance with the latest privacy regulations is validated by a third-party security firm, ensuring that the board and tycoons have verifiable assurance of the system's security measures. |

Table 24: Scenario for Security - Privacy and Data Protection

**Scenarios** In the Deployment Viewpoint, the TRiP system's architecture is strategically distributed across a combination of cloud-based and localized servers to optimize for both performance and security, critical attributes for the robustness and reliability of the system.

The TRiP system's deployment on cloud servers is engineered to enhance performance. The public cloud servers provide a scalable Fast Environment for handling dynamic user demands and instantaneous Route Optimization, ensuring that the most efficient paths are always available to the user. This optimization is further supported by Load Balancers, which are essential in managing network traffic by distributing workloads evenly, thereby preventing any server from becoming a bottleneck and ensuring a smooth user experience.

For security, the system employs private cloud servers, where sensitive components such as the Encrypted Account Lookup Database are hosted. The encryption of this database is a crucial security measure, safeguarding personal and travel data against unauthorized access.

# 8 Development Viewpoint

We start discussing the information viewpoint by analyzing stakeholder concerns that need to be addressed. We individuated the following user stories:

- User stories TODO

As a consequence, we decided to prioritize Quality Attributes for this view as indicated in Table 25.

| | Usability | Performance | Security | Modifiability | Cost Efficiency | Availability | Safety | Integrability | Maintainability |
|---|---|---|---|---|---|---|---|---|---|
| Development View | X | | | X | | | | X | X |

Table 25: Development View Prioritized Quality Attributes

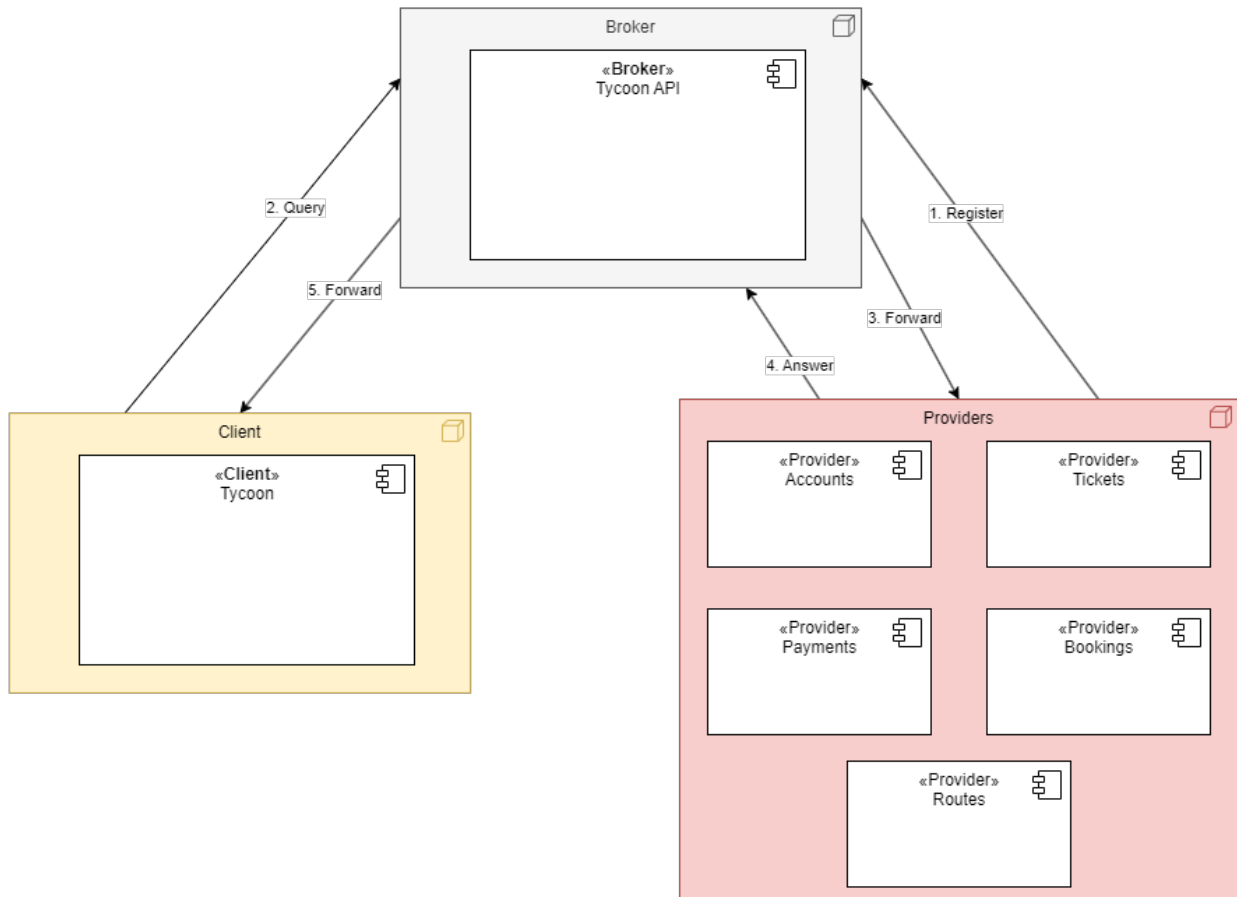## 8.1 View: Broker pattern for Tycoon API

### 8.1.1 Model



Figure 16: Deployment view related broker pattern related to Tycoon API.

### 8.1.2 Description

The diagram portrays the broker architecture in the TrIP system, delineating the communication between clients, the broker, and providers. Tycoon clients initiate the interaction by registering with the Tycoon API broker. Queries are sent from the Tycoon to the broker, which then forwards these queries to the relevant providers. These providers include services for Accounts, Payments, Bookings, Tickets, and Routes, each playing a pivotal role in processing Tycoon requests. Upon gathering the necessary data or performing the required actions, providers send their response back to the broker. The broker, in turn, forwards this information back to the Tycoon client, completing the request-response cycle. This broker pattern facilitates a structured approach to handling requests and centralizes the communication logic, simplifying interactions across the system's diverse components.

### 8.1.3  Glossary of Elements

Table 26: Glossary for the Broker in the Development View.

| Id | Name | Description |
|---|---|---|
| 1 | Client | Represents the Tycoon requesting data or action from the broker. |
| 2 | Broker | Tycoon API that mediates between clients (Tycoon's) and various service providers (TrIP System's modules), handling requests and responses. |
| 3 | Provider | TrIP system that register with the broker to offer their services to clients. |
| 4 | Tycoon API | The application programming interface provided by the broker for communication with clients and providers. |
| 5 | Query | A request sent from the client to the broker, seeking information or action. |
| 6 | Register | The action of a provider adding their services to the broker's list of available resources. |
| 7 | Forward | The process of the broker sending requests to providers or responses back to clients. |
| 8 | Answer | The provider's response to a query that is sent back to the client via the broker. |
| 9 | Accounts | TrIP System module that manages user account information and authentication. |
| 10 | Tickets | TrIP system for issuing tickets, usually related to events or transportation. |
| 11 | Payments | The TrIP module that handles transaction processing. |
| 12 | Bookings | TrIP System module that manages reservations for services offered by the provider. |
| 13 | Routes | TrIP System module that handles route pricing. |

### 8.1.4 Analysis on Perspectives

## 8.2 View: Model-View-Controller pattern for User Interface

### 8.2.1 Model

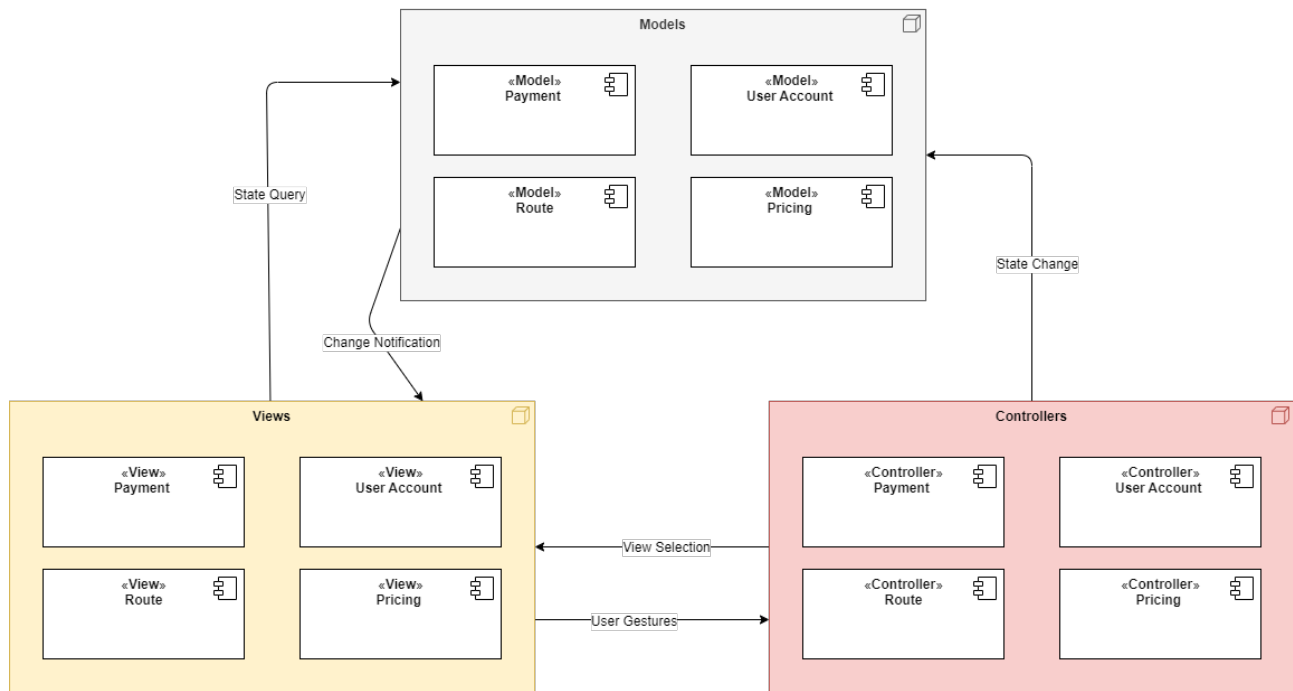**Development Viewpoint - User Interface - Model View Controller**



Figure 17: Development view model-view-controller pattern for UI

### 8.2.2 Description

This diagram illustrates the Model-View-Controller (MVC) pattern applied within the TrIP system. It breaks down the system's user interface into Views, which are responsible for presenting data to users and include interfaces for Payment, User Account, Route, and Pricing. The Models act as the data layer with Payment, Route, User Account, and Pricing components that manage the system's state. They receive state queries and broadcast state changes to the Views. Controllers interpret user gestures, direct state changes, and mediate between the Models and Views. This separation of concerns ensures that user interactions are handled efficiently, system data is managed cohesively, and business logic is executed in a controlled manner. The MVC architecture is fundamental to the system's organization, promoting clear delineations between different functionalities and facilitating ease of maintenance and scalability.

### 8.2.3 Glossary of Elements

Table 27: Legend for the User Interface Components in the Development View of the TrIP System.

| Id | Name | Description |
|---|---|---|
| 1 | Views | User interface components of the TrIP system that passengers interact with, including elements like payment, account management, and route selection screens. |
| 2 | Models | Data structures in the TrIP system that encapsulate the core business logic related to payment, user accounts, route information, and pricing. |
| 3 | Controllers | Components within the TrIP system that manage input from passengers, process requests, and produce responses by interacting with Models and updating Views. |
| 4 | Payment View/Model/Controller | Dedicated interfaces, data, and logic for processing passenger payments within the TrIP system, ensuring secure transactions. |
| 5 | User Account View/Model/Controller | Elements responsible for managing passenger profiles, including authentication and handling of personal account details in the TrIP system. |
| 6 | Route View/Model/Controller | Components that handle the presentation, data management, and operational logic of travel routes available to passengers in the TrIP system. |
| 7 | Pricing View/Model/Controller | The set of user interfaces, algorithms, and data that determine and manage fare calculations and display pricing information to passengers. |
| 8 | State Query | A request by the TrIP system to retrieve the current state, such as checking a passenger's account status or ticket validity. |
| 9 | State Change | An update within the TrIP system that alters the current state in response to passenger actions or other operational changes. |
| 10 | Change Notification | A notification mechanism that informs the TrIP system and passengers when a significant change has occurred, such as a payment confirmation or route update. |
| 11 | User Gestures | Interactions by passengers with the TrIP system, like touch or click actions, which are interpreted as commands to navigate or provide input. |
| 12 | View Selection | The process within the TrIP system for choosing the appropriate user interface view to display to passengers based on the current context or request. |

### 8.2.4 Analysis on Perspectives

| Aspect | Details |
|---|---|
| Source | Tycoon's analytical system requesting data for route pricing and passenger account details. |
| Stimulus | The Tycoon system sends a data request to analyze route prices based on their subscription models and to gather specific account information for passengers. |
| Artifact | Tycoon API interfacing between the Tycoon's systems and the TrIP system's databases. |
| Response | The Tycoon API promptly routes the queries to the appropriate data providers, retrieves the requested information, and consolidates the results for the Tycoon system. |
| Measure | Tycoon system receives comprehensive data within 3 seconds, enabling effective route and subscription analysis for passenger accounts. |

Table 28: Scenario for Tycoon API Data Retrieval

| Aspect | Details |
|---|---|
| Source | TrIP system development team. |
| Stimulus | The need to update the route pricing algorithm without affecting the user interface. |
| Artifact | Model-View-Controller (MVC) pattern within the TrIP system. |
| Response | Developers update the Pricing Model with the new algorithm, which automatically propagates changes to the relevant Views without requiring additional adjustments. |
| Measure | The system update is deployed with zero downtime, and subsequent feature updates require 40% less time due to decoupled architecture. |

Table 29: Scenario for Maintainability - MVC Pattern

| Aspect | Details |
|---|---|
| Source | Passenger using the TrIP system to manage travel. |
| Stimulus | The passenger updates their travel preferences in the User Account View. |
| Artifact | MVC pattern facilitating the User Account management. |
| Response | The Controller processes the passenger's input, updates the User Account Model, and the View dynamically reflects the changes, providing immediate visual confirmation to the passenger. |
| Measure | Passengers express a 90% satisfaction rate with the ease of updating preferences, and the error rate in preference updates decreases by 50%. |

Table 30: Scenario for Usability - MVC-Driven UI

**Scenarios**  The Development Viewpoint focuses on the structural aspects that underpin the TrIP system's architecture, highlighting the modular approach in system design to enhance usability and maintainability.

**MVC Pattern:**
The adoption of the Model-View-Controller (MVC) pattern substantially improves the system's maintainability by segregating the user interface elements (Views) from the business logic (Models) and user input processing (Controllers). This separation of concerns allows developers to update or modify one aspect of the system — such as adding new features to the user interface — without the need to alter the underlying business logic or control flow. For passengers, this results in a more comprehensive interface with a variety of options and features, enhancing the overall usability of the system. The MVC pattern ensures that as passenger needs evolve, new Views can be added swiftly, making the system both adaptable and future-proof.

**Broker Architecture:**
In parallel, the broker architecture employed within the Tycoon API streamlines the interaction process between the Tycoons and the TrIP system. It serves as a centralized point for processing queries and handling communications, thus simplifying the complexity of these interactions. For the Tycoons, this architecture means usability is significantly enhanced, allowing them to communicate through a unified interface with the Tycoon API, which abstracts the intricate workings of the TrIP system's service modules. This consolidation of communication logic not only benefits the Tycoons but also reduces the overhead for the TrIP system in managing multiple communication protocols.

These architectural decisions, reflected in the Development Viewpoint, are key to the TrIP system's ability to offer an efficient, secure, and user-friendly platform that can grow and adapt in line with technological advancements and user expectations.

# A   User Stories

## A.1   Prioritized User Stories

- **User Story 1:** As a frequent traveler, I want to subscribe to a comprehensive monthly pass that includes all three networks so that I can save money on my regular commutes.**QAs:** Usability

- **User Story 2:** As a passenger, I want to be able to check the balance of my multi-network travel card online so that I can easily manage my travel expenses.**QAs:** Usability

- **User Story 13:** As a family, we want to charge train cards from the terminal, so that we can travel when we need without one transaction every time.**QAs:** Usability(+), Modifiability(-)

- **User Story 15:** As an occasional passenger, I want the system to not allow me to pay for temporarily blocked routes, so that I don't have to go through customer service.**QAs:** Usability(+), Modifiability(-), Performance(-)

- **User Story 16:** As a passenger, I want to be able to purchase a single ticket that allows me to travel across all train networks so that I can travel seamlessly without needing to buy separate tickets for each network.**QAs:** Usability(+), Modifiability(-), Performance(-)

- **User Story 18:** As a government regulator, I want the payment system to adhere to data protection and financial transaction security standards so that passengers' personal and financial information is secure.**QAs:** Security(+), Modifiability(-), Testability(-), Integrability(-)

- **User Story 23:** As a tycoon, I want the payment system to integrate with my existing infrastructure with minimal disruption so that I can maintain high service levels during the transition.**QAs:** Integrability(+), Modifiability(-), Usability(+), Testability(-)

- **User Story 26:** As a maintenance team member, I want the system to allow for temporary fare adjustments or bypass options during maintenance work so that we can minimize inconvenience to passengers.**QAs:** Usability(+), Performance(-)

- **User Story 27:** As a network maintenance planner, I want the payment system to integrate with maintenance scheduling tools so that I can plan work with minimal disruption to the train service and revenue.**QAs:** Availability(+), Modifiability(-)

## A.2  Passengers

- **User Story 1:** As a frequent traveler, I want to subscribe to a comprehensive monthly pass that includes all three networks so that I can save money on my regular commutes.**QAs:** Usability

- **User Story 2:** As a passenger, I want to be able to check the balance of my multi-network travel card online so that I can easily manage my travel expenses.**QAs:** Usability

- **User Story 3:** As a passenger with a subscription, I want to receive automatic notifications about my subscription renewal and any discounts or promotions available so that I can take advantage of cost savings.**QAs:** Usability

## A.3  Tech-savvy Passengers

- **User Story 4:** As a tech-savvy passenger, I want to use a mobile app to manage my subscriptions, make payments, and receive digital tickets so that I can have a paperless and convenient travel experience.**QAs:** Usability(+), Deployability(-)

- **User Story 5:** As a passenger interested in sustainability, I want the system to track my travel carbon footprint and offer carbon offset options so that I can make environmentally responsible travel choices.**QAs:** Energy efficiency(+), Performance(-)

## A.4  Accessibility Advocates

- **User Story 6:** As a passenger with mobility challenges, I want the payment system to provide information on accessible services and allow for easy purchase of accessible seating across all networks so that I can travel comfortably and safely.**QAs:** Usability(+), Safety(+)

- **User Story 7:** As an advocate for accessibility, I want the system to offer voice-activated features and screen reader compatibility for passengers with visual impairments so that the service is inclusive and accessible to everyone.**QAs:** Usability(+), Modifiability(-), Deployability(-)

- **User Story 8:** As an elder, I want a simple interface, so that I don't have to get help to buy tickets.**QAs:** Usability(+), Modifiability(+)

- **User Story 9:** As a tourist, I want to choose among many languages, so that I don't have to translate using my phone.**QAs:** Usability(+), Modifiability(-)

- **User Story 10:** As a tourist, I want to have quick access to the most popular trips, so I can quickly get my ticket.**QAs:** Usability(+), Modifiability(-), Performance(-)

## A.5 Passengers with Special Considerations

- **User Story 11:** As a student, I want to be able to scan my student card, so that I can access financial benefits.**QAs:** Usability(+), Modifiability(-)

- **User Story 12:** As a commuting worker/student, I want a single transaction to get all the subscriptions I need, so that I can travel from home to work/school every day.**QAs:** Usability(+), Performance(-)

- **User Story 13:** As a family, we want to charge train cards from the terminal, so that we can travel when we need without one transaction every time.**QAs:** Usability(+), Modifiability(-)

- **User Story 14:** As a visually impaired person, I want a high contrast interface and big font, so that I can avoid mistakes when buying my tickets.**QAs:** Usability(+)

- **User Story 15:** As an occasional passenger, I want the system to not allow me to pay for temporarily blocked routes, so that I don't have to go through customer service.**QAs:** Usability(+), Modifiability(-), Performance(-)

- **User Story 16:** As a passenger, I want to be able to purchase a single ticket that allows me to travel across all train networks so I can travel seamlessly without needing to buy separate tickets for each network.**QAs:** Usability(+), Modifiability(-), Performance(-)

- **User Story 17:** As a student/commuter, I want the system to tell me which subscriptions I need to go from home to school/work, so that I can get the best price.**QAs:** Usability(+), Performance(-), Modifiability(-)

## A.6 Government

- **User Story 18:** As a government regulator, I want the payment system to adhere to data protection and financial transaction security standards so that passengers' personal and financial information is secure.**QAs:** Security(+), Modifiability(-), Testability(-), Integrability(-)

- **User Story 19:** As a government entity, I want the system to provide detailed reporting on passenger numbers and revenue for each network so that we can assess the public transportation system's efficiency and fairness.**QAs:** Usability(+), Modifiability(-), Integrability(-)

- **User Story 20:** As a local government official, I want to ensure that the payment system includes options for reduced fares for eligible populations (students, elderly, low-income) across all networks so that public transportation is accessible to everyone. Also, as a local government, I want to get usage data from the system, so that I can do urban planning properly.**QAs:** Usability(+), Performance(-), Modifiability(-)

## A.7 Tycoons (Train Company Owners)

- **User Story 21:** As a tycoon, I want the payment system to accurately track revenue from shared and exclusive segments of my network so that revenue sharing among tycoons is fair and transparent.**QAs:** Usability(+), Performance(-), Modifiability(-)

- **User Story 22:** As a tycoon, I want the ability to offer exclusive promotions and discounts to passengers using my network to encourage loyalty and increase ridership.**QAs:** Usability(+), Modifiability(-)

- **User Story 23:** As a tycoon, I want the payment system to integrate with my existing infrastructure with minimal disruption so that I can maintain high service levels during the transition.**QAs:** Integrability(+), Modifiability(-), Usability(+), Testability(-)

- **User Story 24:** As a tycoon, I want to access analytics and data insights from the payment system to understand passenger behavior and adjust my services accordingly to maximize revenue and improve service.**QAs:** Integrability(+), Modifiability(-)

## A.8 Financial Analysts

- **User Story 25:** As a station staff member, I want a simple and fast way to assist passengers with ticket purchases and inquiries across all networks so that I can provide efficient customer service.**QAs:** Availability(+), Modifiability(-)

## A.9   Station Staff

- **User Story 26:** As a maintenance team member, I want the system to allow for temporary fare adjustments or bypass options during maintenance work so that we can minimize inconvenience to passengers.**QAs:** Usability(+), Modifiability(-)

## A.10   Maintenance Teams

- **User Story 27:** As a network maintenance planner, I want the payment system to integrate with maintenance scheduling tools so that I can plan work with minimal disruption to the train service and revenue.**QAs:** Availability(+), Modifiability(-)

## A.11   Government Financial Auditors

- **User Story 28:** As a government financial auditor, I want the payment system to include audit trails and compliance reporting features so that we can ensure transparency and accountability in revenue management.**QAs:** The need for audit trails and compliance reporting for transparency and accountability.

## A.12   TrIP Owner

- **User Story 41:** As the TrIP owner, I want to use a service-based architecture, so that the system is easy to maintain and update.**QAs:** Modifiability(+), Deployability(+)

- **User Story 29:** As the TrIP owner, I want the system to operate with the lowest possible operational costs, so that it remains cost-effective over time.**QAs:** Cost-effectiveness over time.

- **User Story 30:** As the TrIP owner, I want the system maintenance efforts to be easily scalable, so that costs do not increase disproportionately as we expand.**QAs:** Scalability of maintenance efforts.

- **User Story 31:** As the TrIP owner, I want routine maintenance tasks to be automated, so that manual labor and operational costs are reduced.**QAs:** Automation of routine maintenance tasks.

- **User Story 32:** As the TrIP owner, I want a cost-effective database solution for data management, so that storage and access costs remain low.**QAs:** Cost-effectiveness of data management.

- **User Story 33:** As the TrIP owner, I want an efficient error handling and reporting mechanism, so that system downtime is minimized and maintenance is cost-effective.**QAs:** Efficiency of error handling and reporting.

- **User Story 34:** As the TrIP owner, I want to implement energy-efficient and eco-friendly technologies, so that the system's operational costs are sustainable and environmentally friendly.**QAs:** Sustainability and environmental friendliness of operational technologies.

## A.13   Additional Considerations

- **User Story 35:** As a new tycoon entering the market, I want to easily integrate my services with the TrIP system, so that passengers can quickly benefit from additional travel options.**QAs:** Scalability and maintainability for new tycoons.

- **User Story 36:** As a system administrator, I want the ability to add and configure services from new tycoons without significant downtime, so that the system remains up-to-date and competitive.**QAs:** System's adaptability and competitive edge.

- **User Story 37:** As a passenger, I want to see new transportation options as soon as they become available, so that I can plan my trips with the most comprehensive information.**QAs:** Availability and usability for new transport options.

- **User Story 38:** As a tycoon, I want my data (such as timetables, prices, and updates) to be accurately and promptly reflected in the system.**QAs:** Accuracy and promptness of tycoon data reflection.

- **User Story 39:** As a passenger, I expect ticket validation to work seamlessly, even during network outages, ensuring that my travel is not hindered by technical issues.**QAs:** Seamless ticket validation during network outages.

- **User Story 40:** As a tycoon, I require the system to handle all transactions securely and efficiently, irrespective of the current network status, to maintain service continuity and safeguard revenue.**QAs:** Secure and efficient handling of transactions under any network status.

# B    Stakeholder Priorities by Events

Table 31: Event Updates for Stakeholder Requirements for the TrIP System

| Stakeholder | Initial QA Priorities | Event 1: New Tycoons | Event 2: Unreliable Network | Event 3: Traffic Jams |
|---|---|---|---|---|
| TrIP Owner | Maintainability (2), Operational Costs (2) | Scalability (2) | Availability (2) Maintainability (1) Cost (1) | *No new change* |
| Tycoons | Usability (1), Reliability (2) | *No new change* | *No new change* | Performance (3) |
| Passengers | Usability (2), Security (1) | *No new change* | *No new change* | *No new change* |
| **Event 4: Data Leaks** | | | | |
| TrIP Owner | Security (2) | | | |
| Railroad Tycoons | *No new change* | | | |
| Passengers | *No new change* | | | |