Software Architecture of Train Inter Payment System (TrIP)

Group 04

March 15, 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 user data.

2 Decisions

2.1 Decision 1: User Interface

Status

Accepted.

Architectural Summary

Concern

Passengers require a user interface that is easy to navigate, visually appealing, and provides a seamless experience across different tycoon systems.

Context

In the context of designing an intuitive and engaging user interface for the TrIP system terminals, we face the challenge of selecting guiding principles and frameworks that will shape the user experience and interaction design. The design of the user interface on the terminals involves the passenger's interaction with the system, from querying routes to finalizing ticket purchases. It is an essential component of the system that directly affects user satisfaction and system usability.

Criteria

The decision will be guided by the following criteria:

- Consistency in design to provide a unified look and feel across all terminals.
- Accessibility to ensure the system is usable by all passengers, including those with disabilities.
- Responsiveness so that the interface can adapt to various screen sizes and orientations.
- Ease of maintenance and scalability for future enhancements.
- Alignment with the latest trends in user interface design and technology.

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 IBS'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

Concern

The main concern lies in selecting a database that can efficiently manage complex data relationships, provide high transactional integrity, and scale as needed without compromising on performance or security.

Context

In developing the TrIP system, tasked with unifying the payment and subscription management across three railroad tycoons' operations, 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

- 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.
- Comprehensive security features to safeguard sensitive data.

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:** 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.

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.

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

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 from various tycoon systems and present them based on the user's criteria of price, time, and subscription status.

Context

In the context of enhancing the TrIP system's ability to provide optimized travel options, we face the challenge of efficiently querying multiple tycoon systems to gather route information that aligns 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:

- User-friendly passenger interface.
- Comprehensive and diverse route options.
- Accurate representation of options based on multiple factors.
- 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.

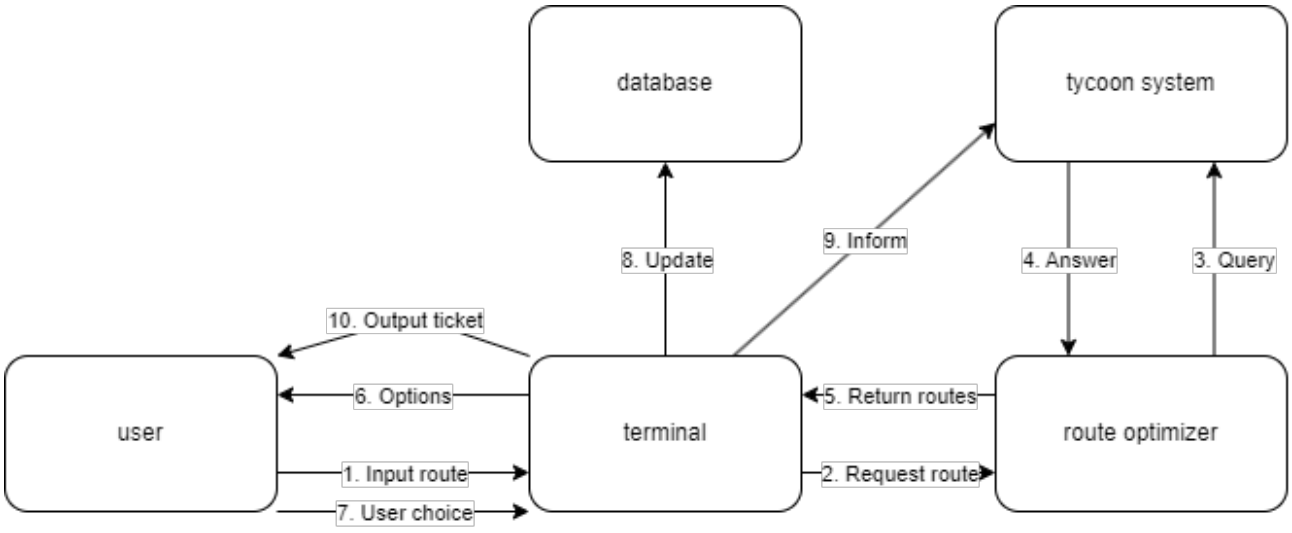

Figure 1: Direct Data Management Interface

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.

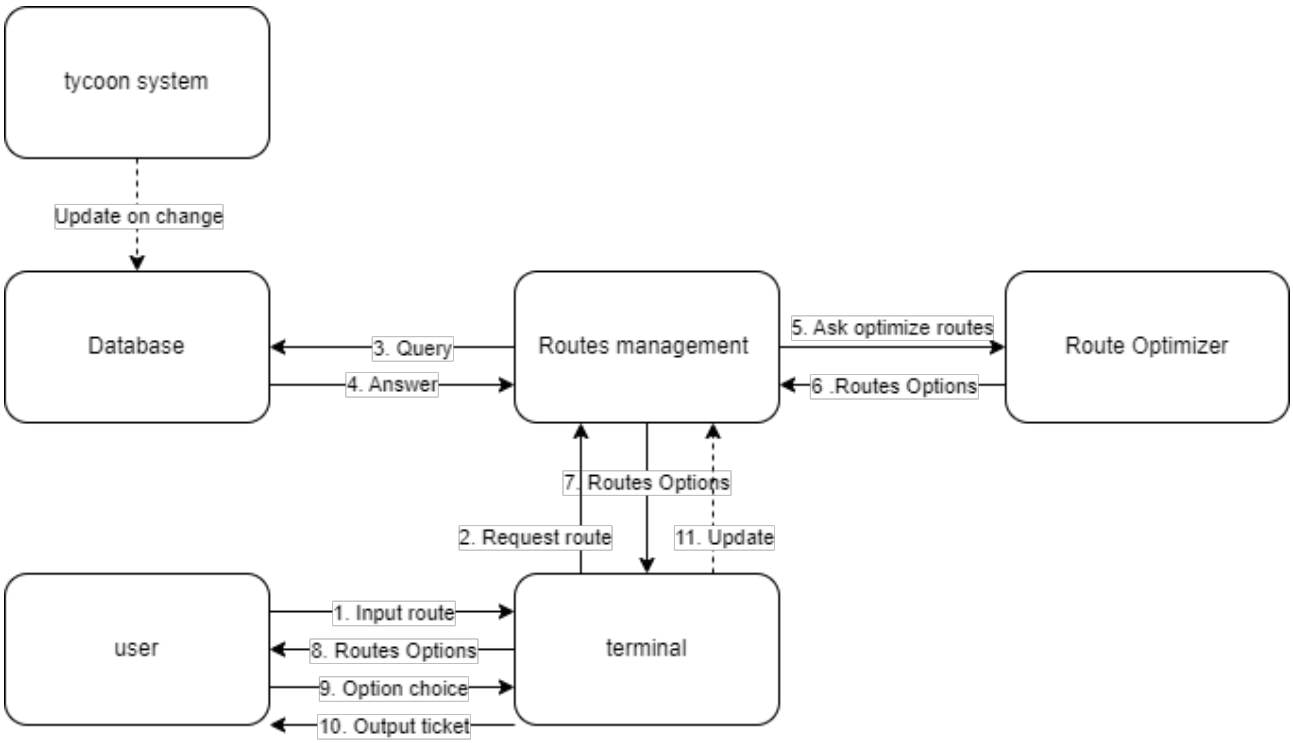

Figure 2: Centralized Route Management Module. Dashed lines represents steps to be explained in future decisions.

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.

2.4 Decision 4: Account and Subscriptions management

Status

Accepted.

Architectural Summary

Concern

A user 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**: 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 tycoon's 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..

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

- Ease of use for the passenger.
- Multiple route options for the user.
- The system returns to user usable options based on price/time/availability/subscription.
- Simple integration with the tycoon systems.

Option 1: Introduction of a subscription manager tool, route management needs to take user 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 users 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 user 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.

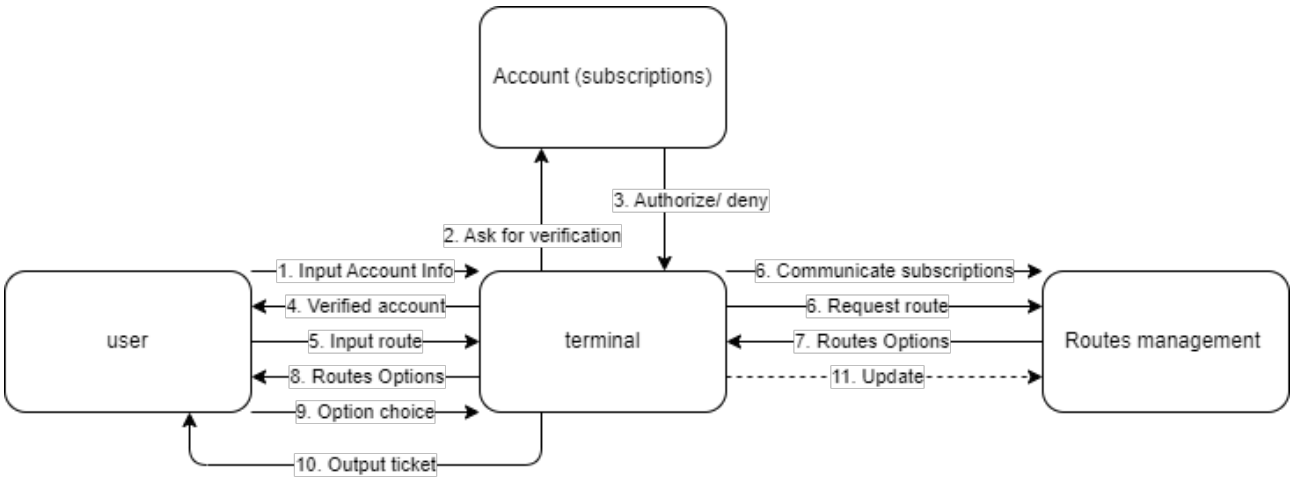

Figure 3: Account Management Interaction for single-ticket.

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 user 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 User Authentication:** Supports various means of user authentication (e.g., magnetic card, smartphone app), offering flexibility and convenience for users to access their subscriptions and travel seamlessly.

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 user via its 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 maintenability, 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

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**: Concerns about paying for unavailable routes.
- **User Story 26**: The need for real-time updates on train schedules.
- **User Story 27**: Integration with maintenance scheduling for minimal service disruption.

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

- Minimize excessive requests to tycoon systems, avoiding system overload.
- Ensure high performance to prevent a negative user experience.
- Prevent multiple users 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 users 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

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 users concerns.

User stories

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 tycoon's 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.

Option 1: Querying modules or external parties should know as little as possible

We try to have one database for each important set of information that the system needs to know. 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.

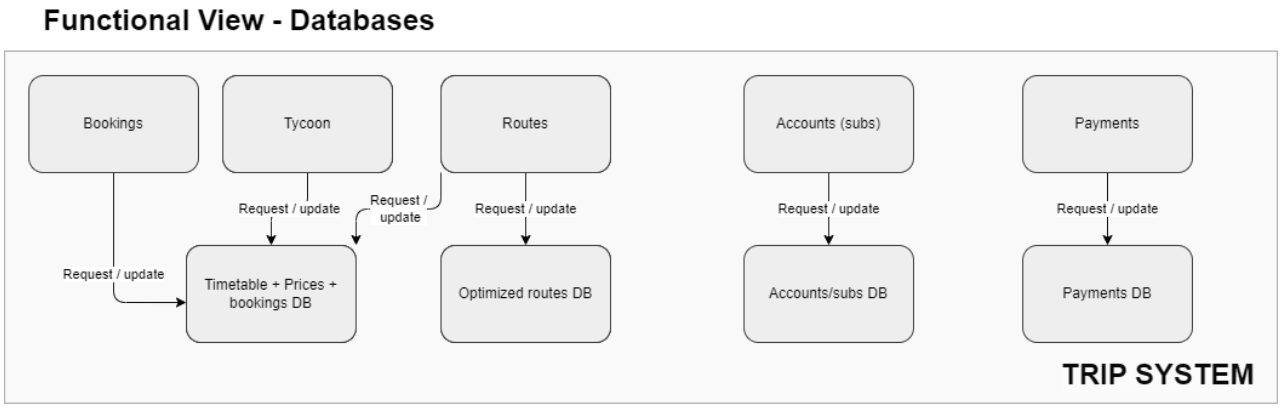

Figure 4: Division of databases and their interaction with modules or stakeholders.

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 specificed 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 or not

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 Hybrid cloud storage System, increased Availability which is important for event 2 which requirees less stringent Maintainability and cost requirements. Cloud increased cost and interaction between databases increase Maintainability compared to single database.

Consequences

Positive Consequences:

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

Negative Consequences:

- **Tycoon permission management** : More complextiy, 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

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.

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

- Timeliness and accuracy of information communicated.
- Data privacy and security compliance.
- Ease of access for customer service representatives.
- Minimization of system complexity and maintenance.
- Integration with existing customer service platforms.
- 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

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 (Frequent Traveler's Monthly Pass)**: A comprehensive monthly pass needs to be recognized by the turnstiles, necessitating a system that supports seamless access for subscribers.
2. **User Story 2 (Online Balance Check for Multi-Network Travel Card)**: The system requires turnstiles to accurately read and verify travel card balances to prevent entry issues, emphasizing the need for an integrated payment system.
3. **User Story 15 (Prevention of Payment for Temporarily Blocked Routes)**: This story underscores the importance of turnstiles having up-to-date information on route availability to avoid mistakenly blocking passengers.
4. **User Story 16 (Single Ticket for All Train Networks)**: Turnstiles must effectively communicate with the payment system to recognize and approve tickets valid across different networks, ensuring seamless travel.
5. **User Story 26 (Real-time Updates for Station Managers)**: The technology supporting real-time updates can also enhance turnstile interactions, ensuring entry permissions reflect current ticketing information based on schedules and disruptions.
6. **User Story 27 (Integration with Maintenance Scheduling)**: Turnstiles must adapt to maintenance schedules, allowing for adjustments in access permissions, which suggests the need for a flexible and responsive payment system.
7. **User Story 23 (Payment System Integration for Tycoons)**: The story relates to the seamless integration of turnstiles with updated payment systems, crucial for maintaining high service levels and secure access.

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

- Allow users to scan tickets or multiplatform travel cards (if they exists) at turnstiles.
- Don't allow people without the appropriate tickets or subscriptions or cards.

- Possibly allow people to scan when they enter and when they exit and calculate a fare (this might depend on agreements between tycoons).
- Possibly communicate station usage data to the system.
- Possibly allow credit/debit card scan.
- Allow balance check for charged cards.

Option 1: Add a ticket checker and a user state modules to authorize user

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.

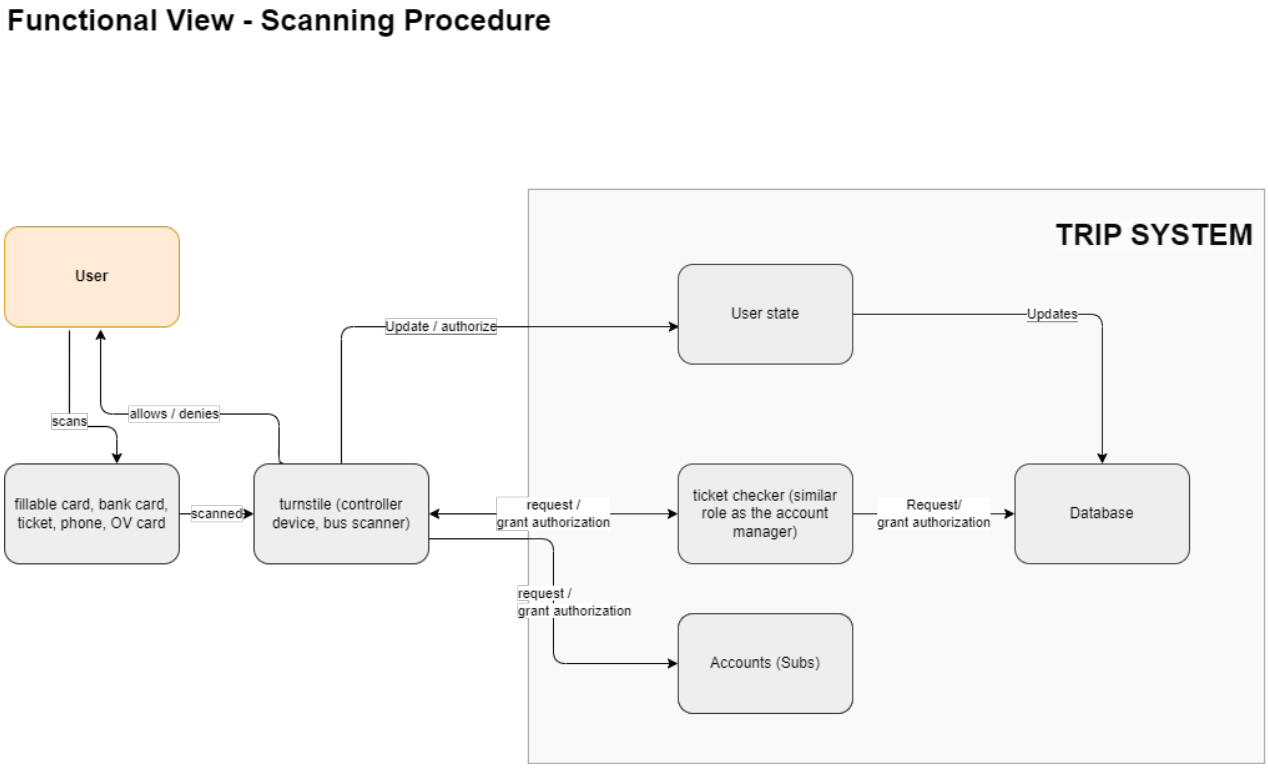

Figure 5: Interaction with a ticket scanner.

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.

Decision

Consequences

Positive Consequences: Negative Consequences:

2.9 Decision 9: Account management and authentication

Status

Review

Architectural Summary

The system will offer multiple options for account management and authentication to accommodate the preferences and needs of diverse user groups, including commuters, tourists, and casual riders. This strategy aims to streamline the user experience while enhancing security and operational efficiency.

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.

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

- Secure storage and handling of personal and financial data.
- Ease of account creation and management for users.
- Seamless integration with existing turnstile and payment systems.
- Flexibility to support different methods of authentication, including cards and digital wallets.
- High availability and reliability of the authentication system.

Option 1: Anonymous card

- **Pro:** Ensures user privacy and quick adoption for casual users.
- **Con:** Limited capabilities for account management and tracking user 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 user 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 user types.

Consequences

Positive Consequences:

- Provides a comprehensive solution covering the needs of various user groups, increasing system accessibility and user 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 users 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 users 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

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

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

- Data correctness in the system, expecially after event 2.
- 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.
- 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 is can be better 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 require a future decision.

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: How to handle disruptions?

Status

Architectural Summary

Concern

User stories

Context

QA Scenario

Usability. Scenario: disruption between A and B.

Criteria

Option 1: Communicate the disruptions and don't do nothing

Option 2: Use google to get alternatives and hybrid routes

Option 3: Calculate alternatives using APIs from other systems (external buses providers)

Decision

Consequences

Positive Consequences: Negative Consequences:

2.12 Decision 12: Disruptions and Route Updates

Status

Open

Architectural Summary

This decision involves establishing a protocol within the TrIP system for managing and mitigating the impact of train disruptions on passengers and service operations.

Concern

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

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.

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

Implement a real-time alert system that notifies passengers of disruptions via mobile app notifications, SMS, and updates on digital displays at stations.

Pros

- Ensures passengers are quickly informed about disruptions.

Cons

- Requires passengers to actively check for updates and take action.

Option 2: Manual Intervention Protocol

Establish a manual intervention protocol where customer service teams are promptly informed of disruptions to assist passengers with rebooking and provide personalized travel advice.

Pros

- Automates the process of managing the effects of disruptions on passengers.

Cons

- May be complex to implement and require significant changes to existing systems.

Option 3: Threshold-Based Re-Optimization

Set a minimum delay threshold. Not rerun the computations if below threshold. If above, rerun optimization. Not inform passengers, all the info is on the terminal. Rerank the best routes based on delays.

Pros

Cons

Option 4: Continuous Optimization

Rerun the computations for every delay. If above, rerun optimization. Not inform passengers, all the info is on the terminal. Maybe rerun computations every hour and tell times might not be correct.

Pros

Cons

Decision

The decision will be made after an in-depth analysis of each option against the set criteria. It will consider the current infrastructure's capabilities, passenger needs, and the potential for seamless integration with other transportation modes. This decision should be merged with decision 11.

Consequences

Positive Consequences: Negative Consequences:

2.13 Decision 13: Serverless vs. Servers for Calculations

Status

Open

Architectural Summary

This decision examines the computational approach for the TrIP system, specifically whether to adopt a serverless architecture or to maintain dedicated servers for processing calculations related to ticketing, route optimization, and other system functionalities.

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.

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-effectiveness, including operational and maintenance costs.
- Performance in terms of latency and throughput.
- Data privacy and control.
- Compliance with data protection and privacy laws.
- Ease of maintenance and updates.

Option 1: Serverless

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

The decision will be based on a cost-benefit analysis of each option, considering the trade-offs between control, cost, and scalability. The chosen approach must align with the overall system requirements for performance, privacy, and regulatory compliance.

Consequences

Positive Consequences: Negative Consequences:

3 Stakeholder Viewpoint

Rationale is missing in this view .

Stakeholder View

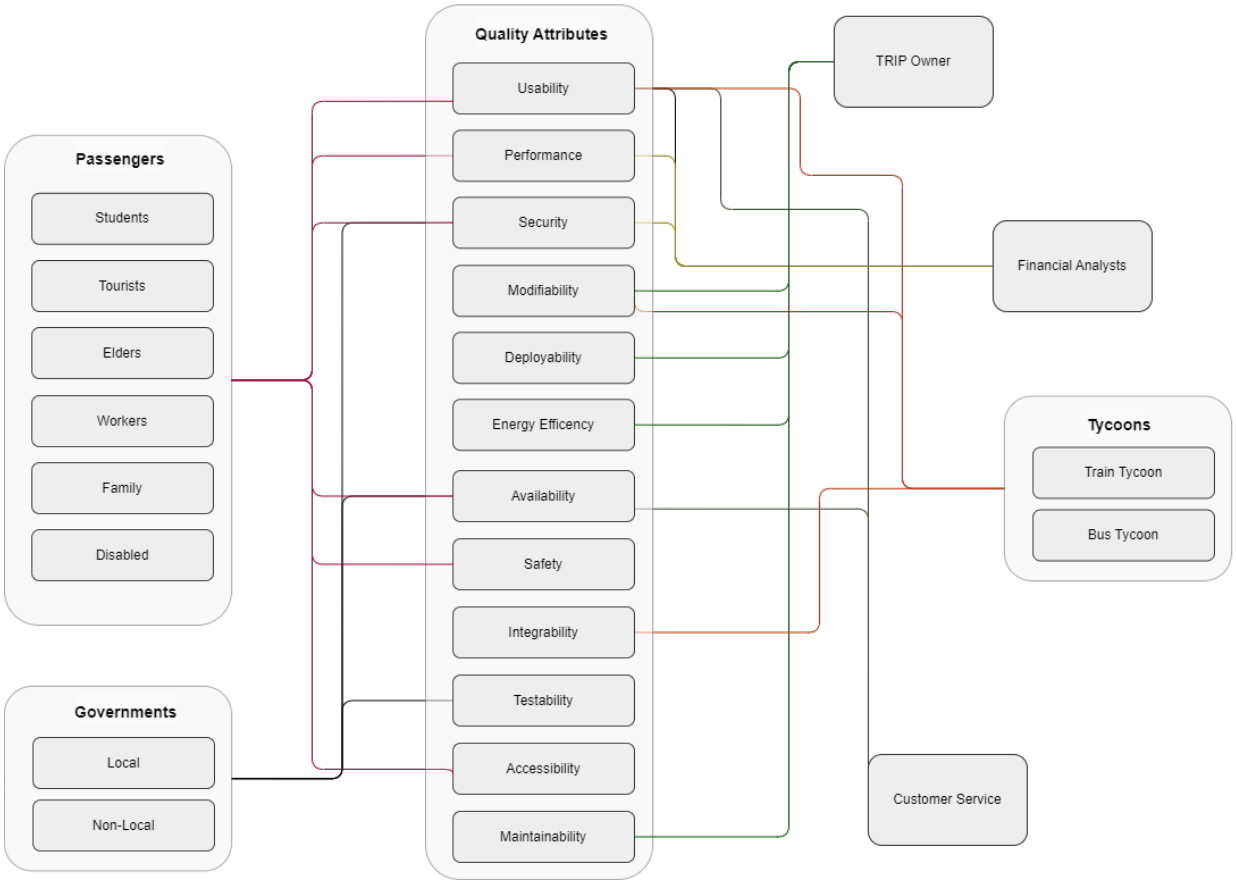

Figure 6: Stakeholder view model.

4 Context Viewpoint

Event 1 is not covered on the context view. According to the grading schema there should be covered in the views that correspond not only in the decisions section.

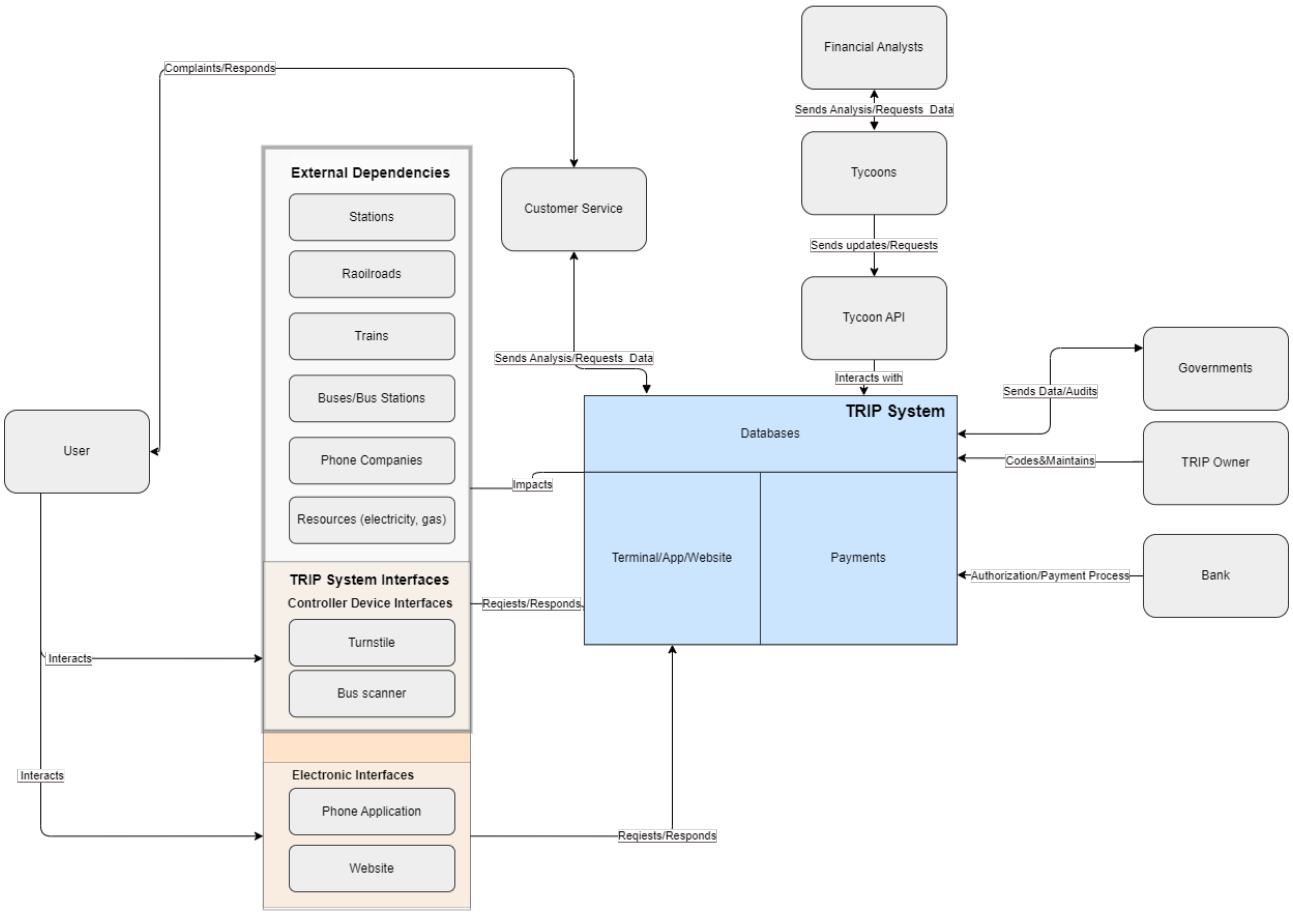

Figure 7: Context view model.

Id	Name	Description
1	User	Individuals who use the TRIP SYSTEM and its associated services, interacting through various interfaces.
2	Customer Service	The department that handles user 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 users, 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 users 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 users interact with for services.

Table 1: Context view glossary for the TRIP System

5 Functional Viewpoint

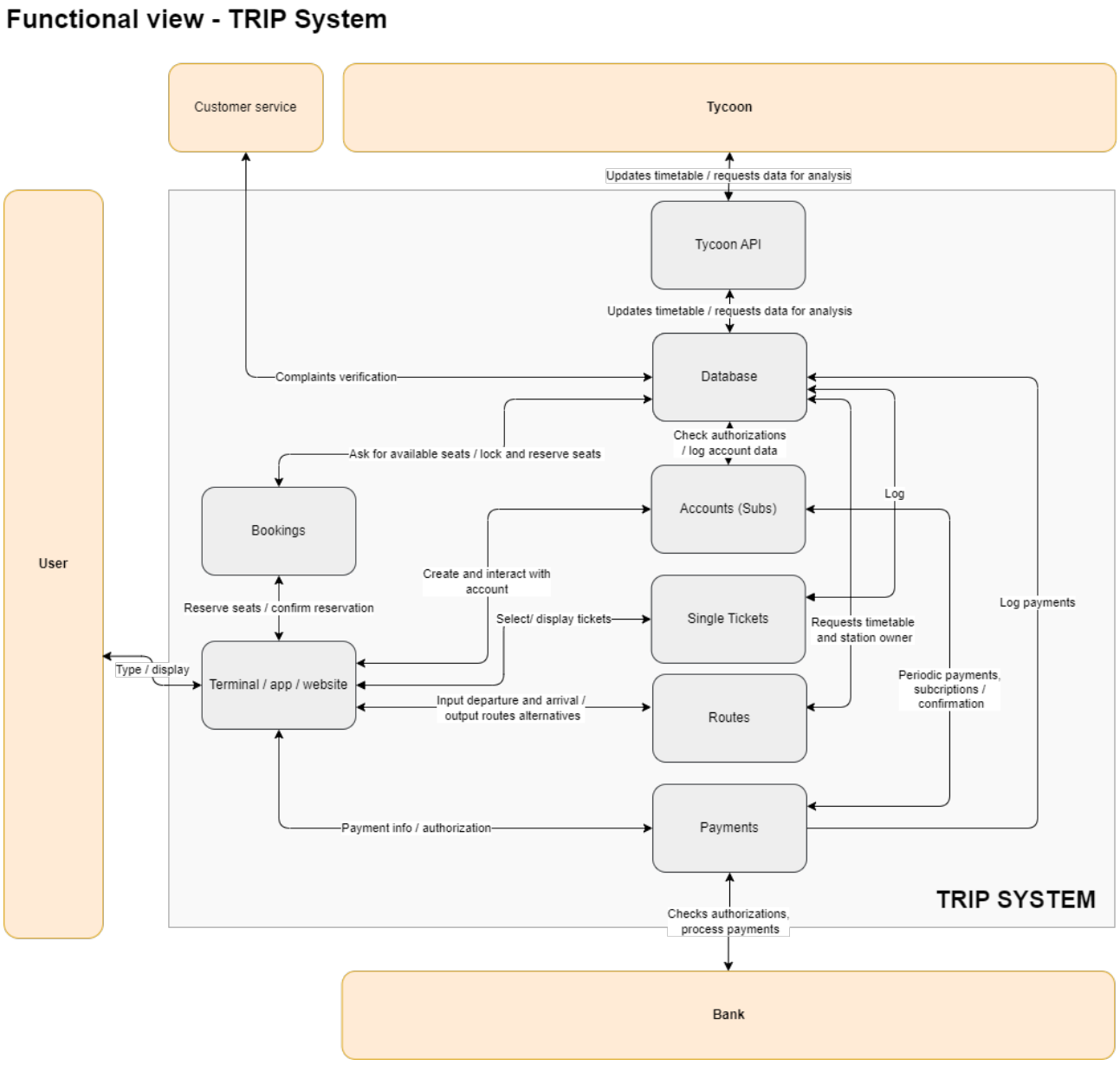

Figure 8: TRIP System.

Id	Name	Description
1	User	End-users of the TRIP SYSTEM who interact with various system components to manage their travel experience.
2	Customer Service	The interface for users 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 user 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 users can inquire about seat availability and make reservations.
6	Accounts (Subs)	The system managing user 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 users with timetables, station ownership details, and route alternatives.
8	Payments	The module handling all financial transactions, including user 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 2: Glossary of elements detailing the components of the TRIP SYSTEM and their roles in facilitating user interaction and service provision.

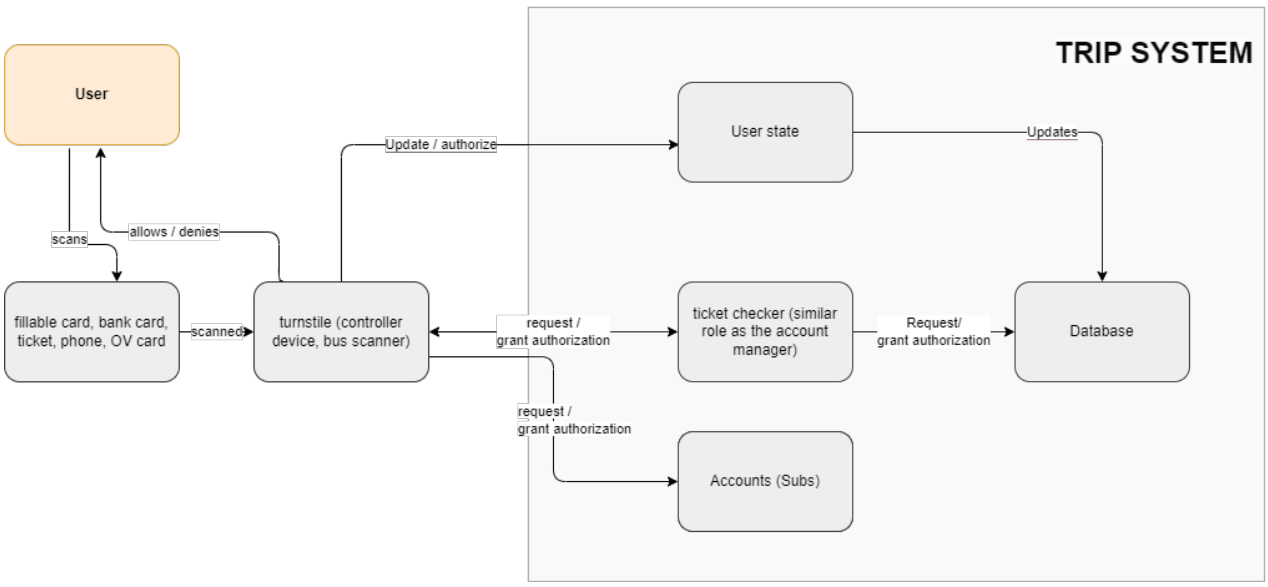

Figure 9: Interaction with a ticket scanner.

Id	Name	Description
1	User	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 user 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 user's ticket or card and determines whether to grant or deny access based on the user state or account information.
4	User State	A system component that maintains the current state of the user within the system, including authorization and access rights, which is updated upon user 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 user access, potentially by checking the user state against the database.
6	Accounts (Subs)	The subsystem managing user accounts and subscriptions, which may interact with the turnstile and ticket checker to verify and update user access rights.
7	Database	An abstraction for the set of databases that stores all system data including user accounts, bookings, and payment information. Detailed information about how different databases are handled is detailed in the Information View.

Table 3: Glossary of elements for the Functional View - Turnstiles, detailing the components and their roles in user access and authorization within the TRIP SYSTEM.

6 Information Viewpoint

Rational is missing in this view. In addition, the impacts of events are not described / explained in the current model.

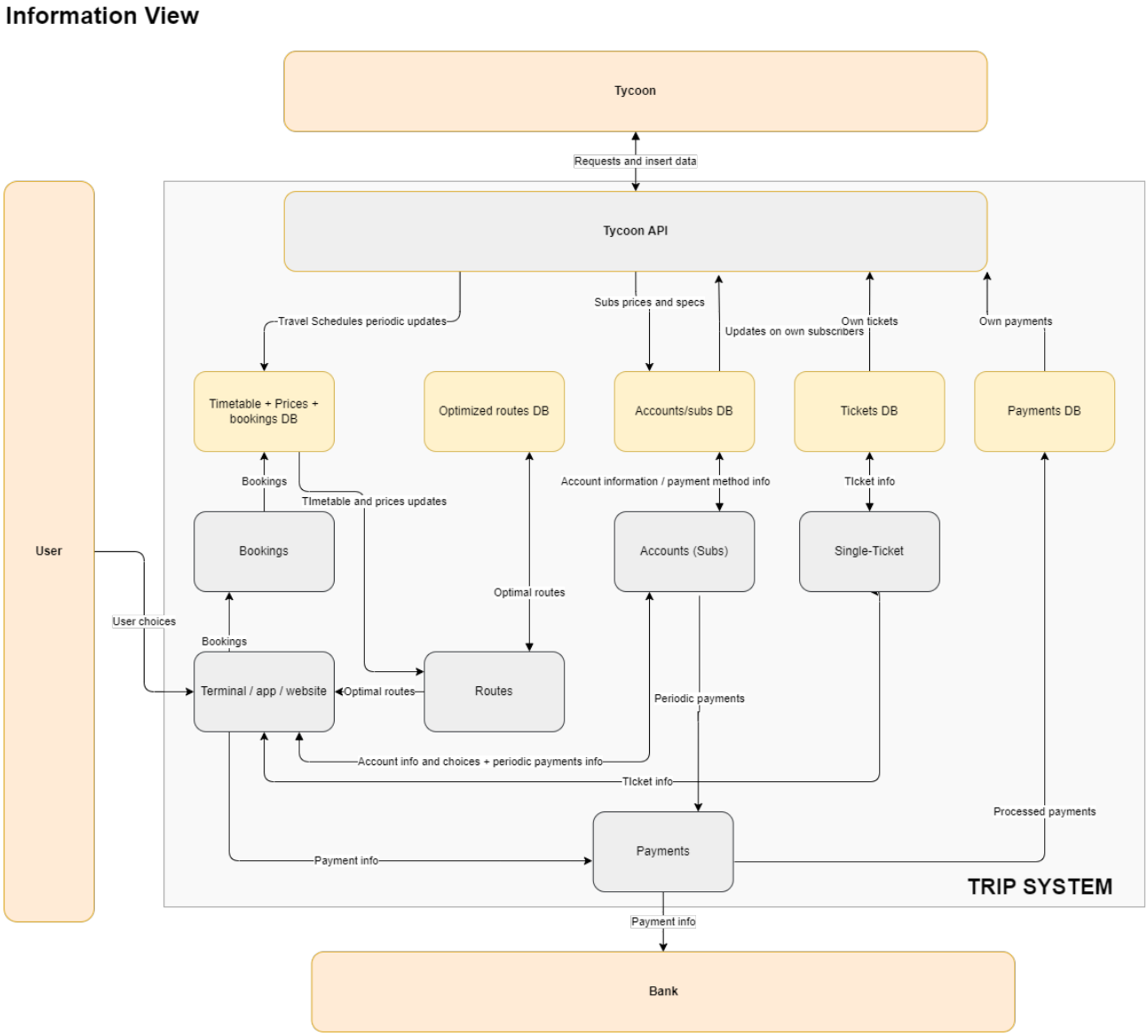

Figure 10: Information view.

Table 4: Legend for the Information View of the TRIP SYSTEM.

Id	Name	Description
1	Timetable + Prices + Bookings DB	The database where travel schedules, prices, and booking details are stored.
2	Bookings	Interface for users to manage their booking requests and view current bookings.
3	Terminal / App / Website	User interfaces for accessing booking, route, and payment information.
4	Routes	Component that contains information about the travel routes and helps in finding the optimal routes.
5	Optimized routes DB	Database storing the most efficient travel routes available.
6	Accounts/Subs DB	Database for managing user accounts and subscription details.
7	Tickets DB	Database where ticket purchases are recorded.
8	Payments DB	Database that records and processes payment transactions.
9	Accounts (Subs)	Management system for user accounts and subscriptions.
10	Single-Ticket	Interface for purchasing individual tickets.
11	Periodic payments	System handling regular subscription fee transactions.
12	Payments	System that processes immediate payment transactions.
13	Processed payments	Record of all completed payment transactions.

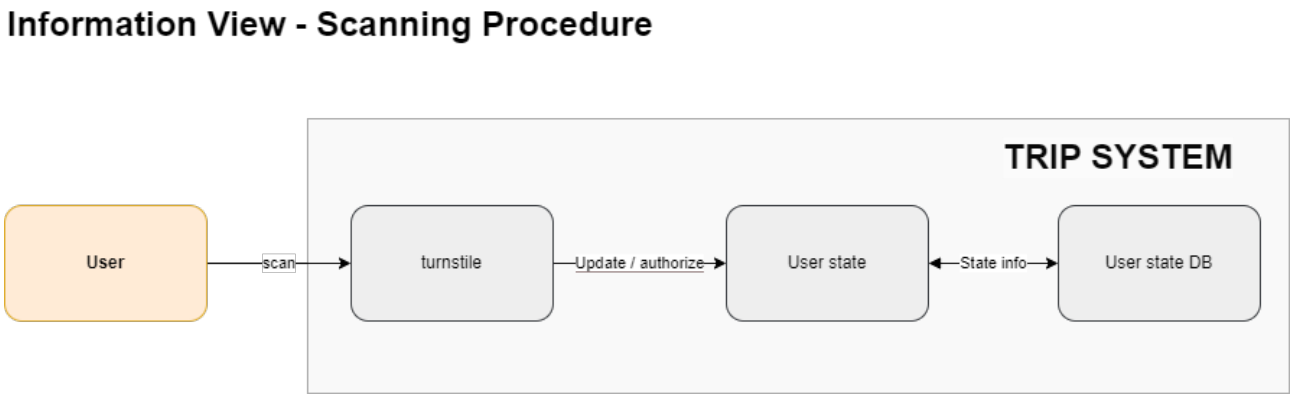

Figure 11: Information view related to the scanning procedure.

Table 5: Legend for the Scanning Procedure in the Information View of the TRIP SYSTEM.

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

Concurrency Viewpoint

Development Viewpoint

Deployment Viewpoint

A User Stories

it will be great to have a brief explanation on how user stories were prioritized.

A.1 Prioritized User Stories

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
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
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(-)
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(-)
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 tycoon's network. **QAs:** Usability(+), Modifiability(-), Performance(-)
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(-)
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(-)
26. As a station manager, I want the system to offer real-time updates on train schedules and network disruptions so that I can keep passengers informed and manage station flow effectively. **QAs:** Usability(+), Performance(-)
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

1. As a frequent traveler, I want
so that I can save money on my
2. As a passenger, I want to be
easily manage my travel expens
3. As a passenger with a subscr
and any discounts or promotion

The provided user stories cover a wide range of stakeholders involved in the trip payment system, from passengers to government entities to train company owners and financial analysts. While most stories prioritize usability, some also address important concerns like accessibility, sustainability, security, and integration. However, considerations for modifiability, deployability, performance, and testability should also be kept in mind and be reflected in user stories to ensure the overall effectiveness and adaptability of the system.

A.3 Tech-savvy Passengers

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(-)
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

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

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(-)
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(-)
11. As a student, I want to be able to scan my student card, so that I can access financial benefits. **QAs:** Usability(+), Modifiability(-)
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 every day. **QAs:** Usability(+), Performance(-)
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(-)
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(+)
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(-)
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 tycoon's network. **QAs:** Usability(+), Modifiability(-), Performance(-)
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.5 Government

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(-)
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(-)
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.6 Tycoons (Train Company Owners)

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(-)
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(-)
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(-)
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.7 Financial Analysts

As a financial analyst for a tycoon, I want the system to provide detailed financial analytics and forecasting tools so that we can optimize pricing strategies and revenue across the shared network. **QAs:** Availability(+), Modifiability(-)

Station Staff

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:** Usability(+), Modifiability(-)

A.8 Maintenance Teams

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:** Availability(+), Modifiability(-)
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.9 Government Financial Auditors

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.