SOFTWARE ARCHITECTURE OF TRIP

Group	05
	Sven Goeseije
	Björn Koemans
	Ravan Strijker

PRODUCT INTRODUCTION

The Train Inter Payment System, also called TrIP, will be a new and innovative way of seamless travel by train or bus. TrIP aims to revolutionize the payment interaction required by passengers by providing a single method to travel all trajectories maintained by various railway or bus operators. The goal of introducing TrIP, is to enhance the travel experience for passengers, regardless of the operator they are depending on. In developing this system, we must address the concerns of passengers, public transport operators and the owner of TrIP.

STAKEHOLDERS

TYCOON

During the development of TrIP there are railway and bus tycoons or operators that will utilize the overarching infrastructures of TrIP. These tycoons are the main stakeholders for developing and operating the system and together they have agreed that the current method of payment is suboptimal for its mainly shared customer base. Therefore, they want to create one shared payment system that supports their individual operations while working seamlessly for passengers. The public transport tycoons are adamant about preserving their traditional fare system and require proof-of-payment verification from passengers. Additionally, they want to maintain their ownership of the operated train stations and bus stops and have the flexibility to add and remove payment terminals at these stations. The primary concern revolves around the reliability, availability and usability of the payment system. As the project's initiator and primary funding source, they hold the greatest interest in TrIP’s success.

PASSENGER

The second main stakeholder for the TrIP project is the passenger who makes use of the services provided by the tycoons. The passengers will use the payment terminals to check in and out of the system every day and are thus the end-users of the overall system. The current system that is in place is too cumbersome and user-unfriendly. The primary concern of the passengers is the ease of use and the security of the system. For example, it is undesirable for the passenger’s personal information, payment information and travelling behaviors to be shared with third parties or become publicly available.

TRIP INC.

TrIP Inc. is responsible for the development and maintenance of the newly introduced payment system. The organization manages everyday operations and ensures the correct handling of passenger payments. Although the initiative for the new system is a collaboration between the public transport tycoons, TrIP Inc. is an independently founded organization which operates in the best interests of both the tycoons and the passengers. The main concern for TrIP Inc. is the maintainability of the system and the costs to operate.

TODO and Notes

Event 1:

Unfortunately, one of the railroad tycoons went bankrupt. To maintain a healthy financial basis, the TrIP *owners decided to broaden their horizon. Fortunately, a bus tycoon stepped in. Therefore, the* TrIP *owners want to open up the platform for other (transport) tycoons as well. The platform should allow for easy tycoon management, so that new tycoons can be added without any development effort. This desired set of functionality is commonly referred to as tenancy.*

Consequences:

TrIP *Owner:*

The QA property for scalability is set to 2: the owner needs to easily connect the TrIP system with the software system of new tycoons.

Changes:

Maken decisions over hoe payment terminals werken: hoe wordt credit bijgehouden + waar worden de transacties opgeslagen.

Railway Tycoons will be addressed as Transport Operators
Train Passengers will be addressed as Passengers
Train Rides will be called Rides
Train Stations will be called Stations and Stops

Train and bus will be called Mode of transport

Verander alle verkeerde trips naar TrIP → done

Decisions

Ravan:
- ERD
- Dataflow+storage between passenger and system
- Dataflow+storage between passenger, payment terminal and system
- Dataflow+storage between tycoon and system
- Shared routes -> money is split between tycoons decision
- Fares decision (simplistic version stored in TrIP system, managed by Tycoons)
- Add fare management to functional view (linked to Check-in validation and Tycoons)
- Maybe add subscription definitions that we allow to Context view, would also be a decision

Bjorn:
Local NFC storage + run-over token each day to prevent chipcard copying
Bus GPS location usage
Conducteurs en fining
Concurrency

Sven:
Use case afmaken
Alles cloud

- **Cloud service model (and where to implement it)**

    - **Public/open/hybrid cloud choice:**
    - **VM: niet gebruiken**
    - **Containers:** gebruiken bij check in and check out data, tycoon monitoring (statistics), tycoon payout and transaction information, add/remove subscriptions/payment terminals/stations/bus stops,
    - **Serverless: niet gebruiken**
    - **Local**
- **Authorization when sending confidential information**
- **Authorization to log into TrIP system for tycoons**
- **GPS for bus check in and out**
- **How does conductor verify proof-of-payment when there is an outtage/ network failure**
    - Alert conductor about outtages, so they dont punnish
    - Add identifier on card, whether they checked in or out that day
    - Checkin or out, station id, timestamp, hashed daily key thing+ internal serial number of chipcard,
    - Automatic rollover private key
- **Fining and conductor**
- **Miss nog iets van customer support achtig iets voor als tycoon niet bij subscription management komt, of passenger niet kan inloggen op TrIP-I account**

Notes from presentation in case notes from other group doesn't cover it:

- Think about authorization
- Add tycoon operators to use case model
- Describe everything you model: statistics is vague (func. view), add arrow from tycoons to payment terminals since they are the ones who manage them
- Also think about how the tycoons discuss subscription prices and such
- Which data is stored where is important, needs to be described in detail
- Route management: done by looking at check in and check out place, then route is calculated and price of route is charged to the passenger
- Context view: add weird drawing on how system works during travel (think of a way how to explain/visualize this) → maybe move the process model to context view
- Think about cost in the same way, how is it calculated and where lies the decision In the systemin

TODO for deadline 4:

- Reflection on handling events
- Customer journey
- Authorization/encryption stuff
- Analysis on perspectives for each model
- Definition of what the Web application does and how it is used by tycoons and passengers

Side notes:

- Table in Cloud approach decision kan beter verplaatst worden naar glossary table bij deployment model denk ik en dan naar verwijzen in de decision.
- Analysis on perspectives voor model is 'Hoe vervult dit model de quality attributes die de stakeholders belangrijk vinden'

DECISION 1: PAYMENT METHOD

STATUS

Open

ARCHITECTURAL SUMMARY

In the context of	Making it possible for passengers to pay for rides
Facing	Compatibility and flexibility
To achieve	A cost-efficient, secure and easy to use payment system
We considered	A paper ticketing solution
	A credit-based chipcard solution
	A debit-card direct payment solution
	Online ticketing
And decided for	A credit-based chipcard solution combined with online ticketing
Because	It provides a straightforward, accessible and risk-free, yet proprietary solution without the need for costly adjustments or involving external stakeholders
Accepting	That users must load credit on their chipcard before travelling, defeating the pay-as-you-go principle The need for a proprietary smartcard

CONCERN

As a passenger, I want to be able to have subscriptions for any of the three tycoons, so that I can travel more economically.

As a tycoon operator, I want to be able to charge passengers for the trip fares, so that I can pay out the tycoon operators the owed money minus a processing fee.

CONTEXT

The decision made impacts the TrIP-I payment processing in the sense of how passengers check in and out at stations or in busses and how the public transport fares are invoiced. This specifically determines how the payment terminals can be used at the stations and busses.

CRITERIA

- Security
- Modularity
- Accessibility
- Modifiability
- Flexibility
- Compatibility

OPTION 1: PAPER TICKETS

Given this option, the passenger must purchase a paper ticket at a ticketing machine at any station for the trajectory he or she is planning to travel on. This is a straightforward approach which is secure and accessible. However, this option is not modular, flexible, or easily modifiable as the goal of introducing TrIP is to make travelling by public transport easier for passengers. This approach requires the passenger to spend a significant amount at a ticketing machine before he or she can start their commute. Additionally, applying subscriptions to this approach is cumbersome and difficult to verify.

Therefore, this approach is simple in implementation and architecture and straightforward in usage with good security and accessibility. However, it is not particularly modular, modifiable, flexible or compatible.

OPTION 2: CREDIT-BASED CHIPCARD

For this approach, every passenger requires a dedicated and proprietary chipcard for travelling. Credit can be ordered and added to the chipcard. This credit is then used to pay for the ride that the user is taking. This approach is less time-consuming than the paper tickets option, but if your credit runs out, new credit has to be ordered and added to the chipcard again which takes time. The credit-based chipcard also allows for the option to couple the existing subscriptions to the tycoon operators onto the chipcard, which makes for an easy transition to the new payment method system. This option is also secure, because the chipcards are coupled to an individual using personal information .

OPTION 3: DEBIT CARD CHECK-IN

The next option is to use a debit card to check-in for a train ride. This option does not require the user to get an extra chipcard which is convenient, making it a very accessible option. Debit card check-in does mean that it is a more anonymized approach than the use of a credit-based chipcard because it is not linked to any personal information. It is therefore less compatible with the existing subscriptions of the tycoons and also less secure. This is why this option would be a great fit for single rides, but not so much for frequent travelers.

OPTION 4: ONLINE TICKETING

Online ticketing is a very accessible and flexible approach. The traveler can get their desired ticket before arriving at the train station or bus stop which is convenient. Instead of waiting in line at a ticket machine you can order a ticket online which saves the traveler a lot of time. However, this option is also less compatible with the existing subscriptions of the tycoon operators. For this option to be compatible with the existing subscriptions, a user must have an account or chipcard in their name to couple the subscriptions to. In combination with the credit-based chipcard this could be used as a flexible, accessible, and modular solution for a payment system.

DECISION

We made the decision to implement features from multiple options. This provides the user with more freedom than just adhering to one of the options to pay for their ticket. The ticketing system and payment method works as follows:

- Passenger has a TrIP-I account
  - Passenger travels with direct bank transfer (SEPA) authorization
    - Personalised Tangible chipcard
    - Personalised Virtual chipcard
  - Passenger travels with credit
    - Personalised Tangible chipcard
    - Personalised Virtual chipcard
  - Passenger travels with subscriptions
    - Personalised Tangible chipcard

        - Personalised Virtual chipcard
- Passenger has NO TrIP-I account
    - Passenger travels with credit
        - Anonymous Tangible chipcard
- Passenger buys single or return day ticket (no TrIP-I required, but optional)
    - One-day-use virtual chipcard (check-in needed)
    - PDF printable ticket (check-in not needed, ticket serves as valid payment method)

In case a user has a TrIP-I account, he or she can request a tangible or virtual (or both) TripCard which is personalized based on the passenger's personal information such as gender, age, name etc. Each personalized TripCard can be used to travel with credit, SEPA authorization or subscriptions. The subscription types are determined by the tycoons, but the passenger subscriptions are managed by TrIP Inc. The passenger selects and pays for the subscription in the TrIP-I environment. Subscriptions must be directly paid for and cannot be invoiced or paid by using credit or SEPA authorizations. However, the subscriptions are priced monthly and passengers may choose the duration for which the subscription should be valid themselves.

When a passenger has no TrIP-I account but wants to travel with credit, the passenger may request an anonymous TripCard. All TripCards have a serial number which is noted on the card, as well as a QR-code which includes this serial number. Upon scanning the QR-code with any internet-enabled device, the user is brought to the TrIP-Anonym-I environment where the passenger can upgrade his or her credit using a multitude of payment options (iDeal, Creditcard, PayPal etc.). Anonymous passengers cannot request a virtual TripCard as digital copies are allowed, meaning that passengers can have their virtual TripCard on multiple digital devices. When an anonymous TripCard would exist, more than one passenger could travel the same train at the same time by sharing a single checked-in TripCard. Therefore, only tangible TripCards are provided for users who wish to stay anonymous. The same reasoning holds for the inability to travel with an anonymous TripCard using subscriptions, as the anonymous TripCard can be shared which leads to a single paid-for subscription being used by more than one passenger.

When a user does not have a TrIP-I account and no anonymous TripCard but wants to travel a certain trajectory (single or return), he or she can purchase a day ticket, which is valid for the entire day of choice. The passenger can choose for a one-day-use virtual TripCard to check in and out with or a printable PDF QR-ticket. The printable PDF ticket cannot be used to check in and out at the stations, but the printed ticket suffices as a legitimate payment method upon random ticket control by conductors.

CONSEQUENCES

The positive consequences of the decision we made are flexibility, accessibility and compatibility. The system allows for various payment options. People that travel a lot can travel with a subscription, which is way more cost-efficient and less time-consuming than paying for a ticket every time you have to take the train or bus. However, for the infrequent traveler, there is an option to only buy individual tickets which makes for a flexible system. The negatives of providing various options to the users are that this will be more costly to implement than just sticking to one option. We do think that the flexible option we came up with is worth it due to the convenience that it provides for the user.

DECISION 2: TRIP MANAGES SUBSCRIPTIONS

STATUS

Open

ARCHITECTURAL SUMMARY

In the context of	Managing subscriptions
Facing	Maintainability and Usability
To achieve	A flexible system to manage changeable subscriptions
We considered	Tycoons managing their own subscriptions
	Centralized subscription management
And decided for	Centralized subscription management
Because	Having a centralized management system for all subscriptions improves ease of use for customers and requires less effort for the tycoons.
Accepting	Higher implementation and integration complexity

CONCERN

As a passenger, I want to easily see all my active subscriptions with any tycoon operator in one place, so that I do not have to retrieve my subscriptions from multiple environments (i.e. tycoons).

As a passenger, I want to see all possible subscriptions for my most used trajectories in one place, so that I can easily compare and choose subscriptions from different tycoons.

As a tycoon operator, I want to be able to determine what kind of subscriptions I provide to passengers, so that I can apply my business strategies.

CONTEXT

The decision impacts the TrIP-I environment in the sense that a subscriptions portal must be added to the passenger's account where he or she can manage active subscriptions and compare optional subscriptions. Additionally, a channel of communication with the tycoons must be set up as the offered subscriptions and the paid-for subscriptions by passengers should be communicated back and forth.

CRITERIA

- Usability
- Maintainability
- Availability
- Security
- Integrity
- Flexibility

OPTION 1: TYCOONS MANAGE THEIR OWN SUBSCRIPTIONS

In case the tycoons manage the subscriptions individually, it means that all passengers must manage and pay for their subscriptions in three separate portals or environments of the different tycoons. It allows the tycoons to easily change their subscription types, the duration of subscriptions and pricing. This also leads to the tycoons overseeing all the relevant handling of processes related to buying subscriptions, such as the payment process and subscription registration.

For this option to work, all tycoons need to individually keep the personal and payment information of passengers, which could cause problems when this data is exchanged with TrIP Inc. when a passenger checks in and it must be checked whether the passenger in question has an active subscription or not. For example, spelling errors in the name of the customer, varying used email addresses, passengers with similar names and so on. On the other hand, the tycoons have more freedom in terms of their offering of subscriptions and their software architecture to provide and manage the subscriptions.

To conclude, while flexibility for the tycoons is valuable, it significantly reduces the usability. Additionally, the individual systems of the tycoons decrease the maintainability of the entire software landscape, as TrIP has to exchange data with the tycoons and thus create integrations with multiple systems. Also, TrIP cannot guarantee the availability of its system as it depends on the systems of the tycoons as well. Lastly, multiple tycoons are in the possession of personal data of passengers, which might introduce security drawbacks.

OPTION 2: TRIP MANAGES ALL SUBSCRIPTIONS

When TrIP manages the subscriptions, passengers must deal with one party only, TrIP Inc. itself. The customer can view all different subscriptions from the tycoons in one place, compare them and pay for them. One of the downsides of this option is the loss of freedom for the tycoons in terms of subscription offerings. The subscriptions must be of some type of category, e.g. particular trajectory, overall discount, or hour of travel. All tycoons must fit their currently existing subscriptions into one of such categories such that these subscriptions can be offered through the TrIP environment. However, this is at the benefit for passengers, who now can more easily view all offered subscriptions and possibly buy subscriptions more often.

This option increases usability, maintainability, availability, and security. TrIP offers one environment for all subscriptions, which is good for usability. The system does not depend on the exchange of subscription information with third parties, which increases maintainability and ensures availability. This option however significantly reduces the flexibility of tycoons. Lastly, the tycoons do not deal with passenger data and payment information, which increases security and data integrity.

DECISION

Criteria/Option	Option 1	Option 2
Usability	-	+
Maintainability	-	+
Availability	-	+
Security	-	+
Integrity	-	+
Flexibility	+	-

We opted for option 2 as we deemed it more viable, where to upsides outweigh the downsides in terms of flexibility.

CONSEQUENCES

TrIP Inc. must invest in the development of an additional TrIP-I environment for passengers where they can view and manage their subscriptions. Additionally, TrIP Inc. has to regularly update the subscription overview and logic, as tycoons remove and add subscriptions to their operations. The tycoons lose the freedom in flexibility to do this themselves. However, the overall costs are smaller than the tycoons managing their own systems plus the costs of building integrations between these individual systems and TrIP. It also reduces the risk of system unavailability due to fewer dependencies, decreasing the risk of big revenue losses for the tycoons.

DECISION 3: ONLINE CREDIT MANAGEMENT

STATUS

Open

ARCHITECTURAL SUMMARY

In the context of	Managing online credit
Facing	Security and accessibility
To achieve	A convenient and secure system that manages online credit
We considered	Vending machines
	Online credit order
	Direct bank transfer (SEPA)
And decided for	A combination of online credit order and direct bank transfers which is managed by TrIP
Because	This adds to the flexibility of the system and prevents users to stand in line at vending machines at train stations
Accepting	Added implementation costs

CONCERN

As a passenger with a chipcard, I would like to manage my credit online so that I do not have to wait in line at the station.

As a passenger with a chipcard, I would like my credit to be secure so that my money is not easily stolen or lost.

As a transport operator, I would like to be guaranteed that passengers pay for their rides so that financial liabilities remain limited.

CONTEXT

The decision impacts the TrIP system and TrIP-I environment in the sense that the user needs to be able to add credit to their chipcard which is safely secured. The TrIP-I and TrIP-Anonym-I environment will need to enable passengers to view and upgrade the credit of their personal or anonymous chipcard if passengers wish to travel credit-based. When passengers want to be directly charged per ride without the need for credit, they must be able to add a SEPA authorization to their chipcard.

CRITERIA

- Usability
- Maintainability
- Availability
- Security
- Modularity
- Flexibility

OPTION 1: VENDING MACHINES

Installing vending machines at each train station to add credit to the TripCards is costly and can lead to waiting in line. On the other hand, vending machines might already be in place at the stations and could be adapted to the TrIP system. Software architecture-wise, this option is the easiest to implement.

OPTION 2: ONLINE CREDIT ORDER

As explained in Decision 1, this option is convenient for users and more cost-efficient for the tycoon operators. Ordering credit online eliminates the time you have to wait in line at train stations for a vending machine and can be done wherever you have internet. This option is easily matched with the chipcards and existing subscriptions of the tycoon operators which makes for great compatibility.

OPTION 3: DIRECT BANK TRANSFER (SEPA)

Direct bank transfers eliminate the need to add credit onto the chipcard. The bank transfers happen automatically if the chipcard owner is using the chipcard to pay for their ride. This saves passengers time since they do not have to check their credit every time they want to get on a train. However, working with SEPA authorizations requires additional financial modules in the software architecture.

DECISION

A combination of online credit order and direct bank transfers which is managed by TrIP is chosen as the final option as described in Decision 1.

CONSEQUENCES

TrIP needs to reach agreements with multiple banks to make sure direct bank transfers can be authorized. They also need to incorporate multiple payment options into their system. By implementing both online credit orders and direct bank transfers, passengers have multiple payment options and can choose the one that suits their situation best.

DECISION 4: STATION MANAGEMENT

STATUS

Open

ARCHITECTURAL SUMMARY

In the context of	Stations and stops
Facing	Accessibility and accountability
To achieve	Seemless integration between the TrIP system and existing stations and routes
We considered	Transport operators managing the stations and stops
	TrIP managing the stations and stops
And decided for	Transport operators managing the stations and stops
Because	This avoids a lot of responsibility and knowledge transfer, and station and stop management is not necessary for the TrIP system to work.
Accepting	Shared management

CONCERN

As a tycoon operator, I want to be able to close or open stations, to adapt to changing passenger travel needs.

CONTEXT

The management of train stations and the train routes these stations are part of is an important responsibility to consider, since it can have huge implications on the TrIP system.

CRITERIA

- Accessibility
- Availability
- Accountability
- Maintainability

OPTION 1:TRANSPORT OPERATORS MANAGING THE STATIONS AND STOPS

Transport operators are already managing the stations, so this option only requires integration with the TrIP system, which needs to be done regardless. By keeping this responsibility with the tycoons, the TrIP system can focus on the more important functionalities, such as subscription integration and payment processing.

This option will keep the necessity of shared stations, requiring more decision-making time in case the tycoons want to change anything.

OPTION 2: TRIP MANAGING THE STATIONS

For TrIP to manage the stations, a lot of responsibility and knowledge transfer will need to happen between TrIP and the tycoons. Additionally, this option requires a large commitment of costs and effort for the TrIP system.

DECISION

The tycoons will keep managing their stations, since this avoids a lot of work that is unnecessary for the TrIP system to work.

CONSEQUENCES

Some stations will still be shared between the tycoons, which requires discussions adding complexity to the decision-making process.

DECISION 5: OWNERSHIP OF PAYMENT TERMINALS

STATUS

Open

ARCHITECTURAL SUMMARY

In the context of	Payment terminals
Facing	Accessibility and accountability
To achieve	An acceptable number of well-located payment terminals at each station and bus.
We considered	Tycoon operators managing payment terminals
	TrIP Inc. managing payment terminals
And decided for	Tycoon operators managing the physical payment terminals
Because	Tycoons already manage the stations and bus tycoons the buses where the payment terminals will be placed, and therefore have more knowledge on the needed amount and locations in each station or bus.
Accepting	Shared ownership

CONCERN

As a transport operator, I want to manage the payment terminals at my stations, so that I can ensure that there are enough payment terminals to handle the amount of passengers.

As a passenger, I want to check in quickly at a payment terminal, so that I do not risk missing my train or bus.

As a bus tycoon, I want the ability to adjust the amount of payment terminals depending on bus capacity and crowdedness.

CONTEXT

The payment terminals are necessary for the TrIP system to work and need to be placed at every station managed by the railway tycoons and bus managed by bus tycoons. It is important to determine who is responsible for placing the payment terminals, since when this is a responsibility of TrIP, it affects the TrIP system.

CRITERIA

- Accessibility
- Availability
- Accountability
- Maintainability

OPTION 1: TRANSPORT OPERATORS MANAGING PAYMENT TERMINALS

In case the tycoon operators manage payment terminals, the initial amount and locations of the terminals per station and bus will likely be the best. Tycoons have the most knowledge about their passenger needs, and therefore have the most information about the numbers of passengers. Another advantage is that the tycoons can easily replace or remove the payment terminals if they have ownership, in case of construction for example.

The main drawback of this option is that some tycoons have shared ownership, and therefore the payment terminals at these stations would also have shared ownership, which will lead to more discussions between the tycoons.

OPTION 2: TRIP MANAGING PAYMENT TERMINALS

If TrIP needs to manage the payment terminals, work is required by all of the stakeholders. There would need to be a lot of communication between TrIP and the tycoons to place new terminals or remove terminals. Furthermore, the TrIP system would need to be expanded to evaluate the efficiency of each terminal.

DECISION

We choose for payment terminal management by the tycoon operators, since this requires less work, discussions and additions to the TrIP system.

CONSEQUENCES

The tycoon operators need to determine how many payment terminals they want to place at each of their stations and where. At shared stations, this will be done by discussing with the other tycoon operators.

DECISION 6: PAYMENT TERMINAL PLACEMENT FOR BUSSES

STATUS

Open

ARCHITECTURAL SUMMARY

In the context of	Payment terminals for busses
Facing	Maintainability, Usability, Availability and Security
To achieve	An easy to use placement of payment terminals for busses
We considered	Payment terminals at bus stops
	Payment terminals inside busses
And decided for	Payment terminals inside busses
Because	This makes for an easier to use terminal for passengers and tycoon operators in the sense that the proof-of-payment is more easily verified.
Accepting	The high implementation costs and temporarily unavailability of busses.

CONCERN

As a bus tycoon operator, I want to have an easily maintainable payment terminal system so I can ensure that the bus rides are paid for by passengers.

As a passenger, I want the payment terminals to be in a convenient place so that I can ensure that paying for my bus ride won't be the reason that I miss the bus.

CONTEXT

The main impacts of these decisions are related to the bus tycoon operators, since the tycoon operators are the ones managing the payment terminals. However, as seen as this decisions impacts the security, maintainability and usability of the system and the tycoon operators are one of the main stakeholders, it is a decision that the use of the system.

CRITERIA

- Usability
- Maintainability
- Availability
- Security

OPTION 1: PAYMENT TERMINALS AT BUSSTOPS

By placing the payment terminals at the bus stops, we can reuse the design placed at train stations. Having them stationary avoids errors in identifying where a passenger checked in or out.

However, this option is costly, since payment terminals would need to be placed at every bus stop. At less popular bus stops, this would be a waste. Since there are almost no conductors in busses, it becomes hard to check whether passengers have checked in or not.

OPTION 2: PAYMENT TERMINALS INSIDE BUSSES

By placing the payment terminals inside busses, the amount of payment terminals used is limited. It is also relatively easy to check whether passengers check in or out, since they have to do this when entering the bus next to the bus driver. This option also protects the payment terminals against vandalism. Since the payment terminals are safely secured inside a bus instead of outside at each bus stop. Passengers do not have to check in before they get onto the bus, which might cause them to miss the bus if they are running late.

The main drawback of this option is that every bus is required to install these payment terminals which is costly causing busses to be temporarily out of commission. Additionally, installing the terminals inside busses requires wireless network connectivity, which is more prone to failure and error.

DECISION

Criteria/Option	Option 1	Option 2
Usability	-	+
Maintainability	-	+
Availability	+	-
Security	-	+

We opted for option 2 as we deemed it more viable, where to upsides outweigh the downsides in terms of usability, maintainability and security. Even though, the costs regarding availability can be higher if multiple busses are out of commission.

CONSEQUENCES

TrIP Inc. must invest in the development of payment terminals for busses, which is costly since the busses will be under construction and therefore temporarily unable to be used.

DECISION 7: CHECK IN HANDLING

STATUS

Open

ARCHITECTURAL SUMMARY

In the context of	Handling check in/out events
Facing	Scalability, Maintainability, Availability
To achieve	An architecture that can handle a highly scalable amount of check in/out events distributed over many payment terminals
We considered	Monolithic architecture
	Microservice architecture
And decided for	Microservice architecture
Because	It offers better scalability and reduces the impact of errors
Accepting	Higher implementation complexity

CONCERN

As the TrIP owner, I want to ensure that the system architecture supports scalability, so that it can handle growing numbers of passengers and transactions without degradation in performance.

As the TrIP owner, I want to create a maintenance-friendly system, allowing for easy updates and bug fixes with minimal downtime.

As a tycoon operator, I want to ensure that all transactions are processed reliably, even during peak times, to maintain passenger satisfaction and trust.

CONTEXT

The TrIP system needs to handle a large and scalable amount of check in/out events, communicated by many payment terminals distributed over different stations and busses. Each time a passenger checks in/out, account and payment information needs to be retrieved to determine how much the passenger should pay for their trip. This has to be done quickly, so passengers do not need to wait at payment terminals for confirmation.

CRITERIA

- Scalability
- Maintainability
- Availability

OPTION 1: MONOLITHIC ARCHITECTURE

In a monolithic architecture, the entire application is built as a single codebase, typically organized into modules or layers. Components within the application are tightly coupled, meaning changes to one part of the application may require modifications to other parts as well.

The entire monolithic application would be deployed as a single unit, simplifying deployment but potentially leading to longer deployment times and increased risk during updates.

Scaling a monolithic application often involves scaling the entire application vertically by adding more resources (e.g. CPU, memory) to the server hosting the application.

OPTION 2: MICROSERVICE ARCHITECTURE

In a microservice architecture, the application is decomposed into smaller, independently deployable services, each responsible for a specific function. These functions would include requesting account information, calculating a due amount and sending confirmation messages to the payment terminals.

Microservices communicate with each other through well-defined APIs, allowing for looser coupling between components and greater flexibility for individual service deployment. Each microservice can be developed, tested and deployed independently, enabling faster release cycles and reducing the risk associated with deploying changes.

Lastly, microservices offer fine-grained scalability, allowing individual services to be scaled independently based on demand. This can result in more efficient resource utilization compared to monolithic architectures.

DECISION

Criteria/Option	Option 1	Option 2
Scalability	-	+
Maintainability	-	+
Availability	+	+

We chose for a microservice architecture, since it complies with the important system qualities scalability and maintainability better than a monolithic architecture.

CONSEQUENCES

Building a microservice architecture results in more effort than a monolithic architecture, but it will be easier to maintain and update.

DECISION 8: PAY TERMINAL COMMUNICATION

STATUS

Open

ARCHITECTURAL SUMMARY

In the context of	Payment terminals communicating with the TrIP system
Facing	Availability, Reliability and Security
To achieve	A fault-tolerant communication service with low latency
We considered	API gateway
	Direct client-to-service communication
And decided for	API gateway
Because	An API gateway is easier to implement and has better security
Accepting	Potentially higher latency

CONCERN

As a tycoon operator, I want to ensure that all transactions are processed reliably, even during peak times, to maintain passenger satisfaction and trust.

As the TrIP owner, I want to implement robust security measures, to protect passenger data and prevent unauthorized access.

CONTEXT

For the TrIP system to handle passengers that check in/out, payment terminals need to communicate these events with the system. To make sure no messages get lost and passengers do not need to wait for too long, the communication service should be fault tolerant and have low latency.

CRITERIA

- Availability
- Reliability
- Security

OPTION 1:API GATEWAY

API gateways provide a centralized entry point for clients to access backend services, simplifying client-server communication and providing a unified interface. They can aggregate and compose responses from multiple backend services, providing a consolidated view for clients.

API gateways can also enforce security policies, handle authentication and authorization, and provide features such as rate limiting and request logging. Requests can be routed to appropriate backend services based on routing rules and perform load balancing to distribute traffic across multiple instances of services.

OPTION 2: DIRECT CLIENT-TO-SERVICE CONNECTION

With direct client-to-service communication, clients communicate directly with individual backend services, eliminating the need for a centralized intermediary. Removing the additional hop through an API gateway can potentially reduce latency in client-server communication.

Direct communication can increase the complexity of client implementations, as clients need to be aware of service endpoints and potentially handle service discovery and routing themselves. Lastly, security mechanisms such as authentication and authorization need to be implemented at the service level, potentially leading to duplication of effort and increased complexity.

DECISION

Criteria/Option	Option 1	Option 2
Availability	+	+
Reliability	-	+
Security	+	-

We opted for an API gateway communication service, because we think the added security benefits outweigh the lower latency compared to direct client-to-service connection. Especially since lower latency is not guaranteed for client-server communication. Furthermore, the complexity of direct client-to-server communication is higher then that of API gateways.

CONSEQUENCES

The communication service will have good security and routing, but latency is potentially higher.

DECISION 9: CLOUD APPROACH

STATUS

Open

ARCHITECTURAL SUMMARY

In the context of	How the TrIP system is deployed
Facing	Availability, Security and Cost concerns
To achieve	A cost-effective and secure deployment model
We considered	Public cloud
	Private cloud
	Hybrid cloud
	Multi-cloud
	On-premise
And decided for	Hybrid cloud TrIP system and On-premise payment terminals
Because	The security private cloud provides in combination with the cost effectiveness of public cloud make for a good solution
Accepting	Possible cloud provider network problems and lower security than an on-premise architecture for the TrIP system

CONCERN

As a passenger, I want to be able to pay for a single fare commute by train, so that I do not risk a fine.

As a passenger, I want my personal data to be shielded from unauthorized access, so that my privacy is protected and my trust in the TrIP system is maintained.

As a transport operator, I want to ensure that all transactions are processed reliably, even during peak times, to maintain passenger satisfaction and trust.

As the TrIP owner, I want to ensure that the system architecture supports scalability, so that it can handle growing numbers of passengers and transactions without degradation in performance.

As the TrIP owner, I want to implement robust security measures, to protect passenger data and prevent unauthorized access.

CONTEXT

Besides what modules the TrIP system has and how they function, how they are deployed is also very important. Communication between the modules and how the system functions when there is a network outage all depends on how the infrastructure is deployed. This decision in combination with the deployment model will elaborate on how the system is deployed.

CRITERIA

- Availability
- Flexibility
- Scalability
- Security
- Cost

OPTION 1:PUBLIC CLOUD

Advantages:

- Available to all
- Cost effective (no hardware or software purchases required + no ongoing maintenance costs)
- Cloud providers have data centers located at different regions across the world (high redundancy)
- Highly scalable

Disadvantages:

- Minimal control (not very customizable, so less flexible)
- Less secure than private cloud as the platform is shared

OPTION 2: PRIVATE CLOUD

Advantages:

- On-premise (hosted privately in your own datacenter, which means that data which needs to be secured by your own standards can be done in your own environment)
- Flexibility
- Scalability
- Highly isolated and secure
- Operates in your chosen cloud or on-premises data center

Disadvantages:

- More costly (upfront costs + operational/management costs → requires in-house DevOps team)
- Dependence on service provider

OPTION 3: HYBRID CLOUD

Advantages:

- Can reap the benefits of both public and private cloud

Disadvantages:

- With various cloud services from different providers, there are additional complexities in maintaining a hybrid cloud. In-house installation, management, and maintenance expenses can quickly add up, and there may be more security risks with data moving between services

OPTION 4: MULTI-CLOUD

Advantages:

- The ability to use various cloud services for different departments
- The flexibility to easily switch between vendors
- Less downtime with strategically located data centers

Disadvantages:

- Migrating to a multi-cloud strategy and managing multiple vendors can be time-intensive and complex. Also, your multi-cloud operations stay siloed, meaning data is not transferred between services. But depending on your needs, this might be a benefit
- There is a need for your IT team to have the knowledge and skill sets to handle maintenance for multiple cloud deployment methods

OPTION 5: ON-PREMISE

Advantages:

- Maximum control
- Performance (faster performance, less latency than cloud solutions)
- Predictable costs
- Security (configure own security)

Disadvantages:

- Maintenance and upgrades (On-premise data centers require regular maintenance and upgrades, which can be time-consuming and expensive)
- Scalability (On-premise data centers are less scalable than private clouds, as businesses must purchase and install additional hardware to increase capacity)
- Cost (On-premise data centers require significant upfront investment in hardware, software licenses, and infrastructure. Additionally, ongoing maintenance and upgrades can be expensive)

DECISION

Criteria/Option	Option 1	Option 2	Option 3	Option 4	Option 5
Availability	+	+	+	+	++
Flexibility	-	+	+	+	++
Scalability	++	++	++	++	--
Security	-	+	+	+	++
Cost	++	-	+	-	--

We opted for a hybrid cloud as the main deployment method, so that we can reap the benefits of both public and private cloud. Public cloud is more cost-effective, but less flexible and secure than private cloud, which is why public cloud is a good deployment method for the functional system components where security is not a main concern. Using hybrid cloud, the functional system components where security is a top priority can utilize a private cloud architecture.

We opted for an on-premise approach for the payment terminals, so that if the network may fail, the transactions are still recorded by the payment terminal itself. This way, passengers will be able to check in when there is a network problem and the transactions aren't lost, so the tycoons can still be paid after the network issue is resolved.

However, an on-premise approach for the whole system is less than ideal. Hybrid cloud is a more flexible and easier scalable approach than on-premise. The cost of maintenance and upgrades to the system and lack of easy scalability does not outweigh the upsides on the security, because private cloud can provide enough security for the modules of the system that require tight security.

An overview of the system's functional components with the chosen cloud approach is shown in the table below. For a full description of each functional system component, see the functional model with accompanying glossary table. For a detailed overview of the deployment of the system, see the deployment model with accompanying glossary table.

System functional components:	**Cloud approach and description**
Payment terminal	**On-premise:** Payment terminals are their own devices managed by the tycoons, only sends info to check-in validation module.
Account management	**Private cloud:** This module handles account information of passengers, including TripCard details, linked subscriptions, added payment methods and current credit, which is personal information that needs tight security.
Subscription management	**Public cloud:** This module handles all the different subscriptions that are defined by the railway tycoons. Security is not that much of a concern, since no sensitive information is involved.
Payment processing	**Private cloud:** This module handles every occurrence where a payment should be made. This is sensitive information and requires tight security.
Check-in validation	**Private cloud:** This module handles every occurrence where a passenger checks in/out at a payment terminal handling information like TripCard details and payments. This is sensitive information and requires tight security.
Monitoring	**Private cloud:** This module gathers monitoring data from both accounts and payments to create statistics. This is sensitive information and requires tight security.
Customer support	**Public cloud:** This module deals with questions from passengers and attempts to answers them. No sensitive information is exchanged meaning that security of public cloud is enough.
Subscription editing	**Public cloud:** This module allows passengers with a TrIP-account to manage their subscriptions. Same as Subscription management, security is not that much of a concern, since no sensitive information is involved.
Payment management	**Private cloud:** This module allows passengers to manage the payment methods for their account. This is sensitive information and requires tight security.
Credit management	**Private cloud:** This module allows passengers to add credit to their account. This is sensitive information and requires tight security.

One-day ticket management	**Private cloud:** This module allows passengers without an account to buy one-day tickets for a specific trip. Since payment are seen as sensitive information, private cloud ensures tight security.
Fare management	**Public cloud:** This module handles the fares determined by the tycoons. The fares are common knowledge and do not require tight security.

CONSEQUENCES

Making sure that the sensitive information is tightly secured using a private cloud architecture means that the costs will be higher. If there is a network problem at the cloud provider than the system will be temporarily unavailable.

DECISION 10: FARE DISTRIBUTION ON SHARED ROUTES

STATUS

Open

ARCHITECTURAL SUMMARY

In the context of	Routes between shared stops
Facing	Maintainability and Operational costs
To achieve	A fair distribution of reveue and effort on routes shared by transport operators
We considered	Splitting revenue based on amount of passengers
	Predetermined distribution of effort and revenue
And decided for	Predetermined distribution of effort and revenue
Because	This simplifies the data model and calculations, avoiding high operational costs
Accepting	Added communication and need for agreements between transport operators

CONCERN

As a transport operator, I want to be able to close or open stations, to adapt to changing passenger travel needs.

As a transport operator, I want to ensure that all transactions are processed reliably, even during peak times, to maintain passenger satisfaction and trust.

As the TrIP owner I want to ensure the system integrates seamlessly with each tycoon's existing infrastructure, for smooth operation and user experience.

CONTEXT

Because stations and bus stops can be shared by transport operators, there can be routes between two stops that are shared by the same transport operators. On these shared routes, the amount of deployed trains/busses per transport operator will likely not be equal. This could lead to difficulties in determining what portion of the fare revenue each transport operator should receive for these routes.

CRITERIA

- Maintainability
- Operational costs

OPTION 1: SPLITTING PROFIT BASED ON EFFORT/AMOUNT OF PASSENGERS

One solution is determining how many trains/busses each transport operator has deployed and how many passengers were on these trains/busses/trams/metros etc., and using this data to determine what share of the revenue should be given to each transport operator. This option requires no extra communication or agreements between transport operators, but has a large impact on the data model and services of the TrIP system.

Solely basing revenue splits on the amount of deployed trains/busses would not be fair, since this would allow transport operators to deploy a large amount of trains/busses that will transport barely anyone. The amount of transported passengers is a much better indication of how revenues should be split.

Determining what vehicle a passenger took on a shared route requires a lot of data to be tracked actively. The system will have to take into account a lot of different aspects, like delays, early check-ins and late check-outs, to determine which transport operator facilitated a passenger's trip.

OPTION 2: PREDETERMINED DISTRIBUTION OF EFFORT AND PROFIT

The other solution is forcing transport operators to agree on a distribution of effort and fare revenue for each shared route. This option requires a lot of communication and discussion between the transport operators, but has a neglectable impact on the data model and services of the TrIP system.

Transport operators are required to agree on the amount of trains/busses each deploys, and the share of revenue each transport operator will receive for that. The TrIP system then only needs to be updated on changes in these agreements to determine what portion of fare revenue should be paid to whom.

DECISION

Criteria/Option	Option 1	Option 2
Maintainability	+	-
Operational costs	--	+

We opted for predetermining the distribution of effort and profit, since this requires barely any extra operation costs for the TrIP system. The amount of shared routes and transport operators should also stay relatively small, so this option should be manageable for the transport operators.

CONSEQUENCES

High operational costs on the data model and services of the TrIP system are avoided, but transport operators will need to make agreements on the distribution of effort and profit on shared routes.

DECISION 11: FARE PRICING

STATUS

Open

ARCHITECTURAL SUMMARY

In the context of	Fare prices between shared stops
Facing	System maintainability and complexity for stops with stationary payment terminals
To achieve	An easy way to distribute fare revenues among transport operators
We considered	Transport operators determine their own fare price, even for shared routes
	Equal fare prices for all transport operators on shared routes
And decided for	Equal fare prices for all transport operators on shared routes
Because	This simplifies the complexity of determining which passenger travelled with which operator, allows for clear communication to passengers and avoids fare fraud
Accepting	Transport operators having to settle for a fare that would possibly be lower if a route is shared compared to when they would set their own prices

CONCERN

As a transport operator, I want to be able to determine the fares for the route I operate, so that I have control over my revenue streams.

As a transport operator, I want to be ensured that I receive the correct fare revenue amount, so that I do not miss any revenue.

As a passenger, I want to know what my trip will cost no matter at what time I choose to travel with any of the operators, so that I am never surprised by variable trip fares.

As a passenger, I want the fare pricing to be straightforward and easy to calculate, so that I can prepare for my trip and know how much credit I need.

As the TrIP owner I want to ensure the system integrates seamlessly with each tycoon's existing infrastructure, for smooth operation and user experience.

CONTEXT

Considering Decision 10 on the fare distribution based on the deployment of vehicles by different transport operators on shared routes, this decision follows. In Decision 10 the choice was made to predetermine the distribution of vehicles and timetables instead of split revenues based on number of passengers. This means that the total fare revenue for a certain day for a certain shared piece of route is based on the transport operator’s share in the entire timetable for that day. However, this brings another problem to the table, as it is still unspecified how fares are calculated on shared routes since operators have control over their own fare tariffs.

CRITERIA

- Maintainability
- Usability

- Flexibility
- Reliability

OPTION 1: TRANSPORT OPERATORS DETERMINE THEIR OWN FARE PRICE, EVEN FOR SHARED ROUTES

This solution allows each operator to determine individual fare tariffs for any given route, even if the route is shared. However, as we decided that fare revenues are paid out based on timetable shares, it would result in having to change the check-in fare prices dynamically throughout the day, depending on which operator is responsible for the next arriving vehicle at a certain stop. This not only makes it confusing for passengers with ever-changing tariffs, it is also immensely complex. When there are variable tariffs, combined with peak hours, payout based on timetable shares, and no data on which passenger travelled with which operator on a track, it is complex to calculate which operator receives what amount.

Additionally, it would require the payment terminals and check-in modules to be aware of the timetable and next arriving operator, where the timetables are not a responsibility of the TrIP architecture. For this to work, additional dependencies must be added to check with the different operator’s schedules.

OPTION 2: EQUAL FARE PRICES FOR ALL TRANSPORT OPERATORS ON SHARED ROUTES

An alternative option is to assume or demand the operators to agree on fare prices on shared routes. In Decision 4 it is decided that transport operators remain the owner of the various stations and stops. It could be assumed that when an operator owns a station and another operator wants to make stops with their vehicles at these stations or stops, the new operator should agree to the existing prices, the existing and new operators settle for new tariffs or the new operator is simply denied.

This results in a more maintainable architecture with relatively stable fares. The check-in modules are also independent of the operator and the passenger is ensured of a stable fare.

DECISION

Criteria/Option	Option 1	Option 2
Maintainability	-	+
Usability	-	+
Flexibility	+	--
Reliability	--	++

We opted for equal fare prices for all transport operators on shared routes. This greatly reduces the system complexity, making it easier to maintain in the future. Also, usability for the passenger is better due to the clearly communicated prices instead of dynamic pricing. For the transport operators, a sense of flexibility is lost as they are not able to determine their own pricing models everywhere. However, the system has become highly reliable due to the fewer dependencies needed.

CONSEQUENCES

Transport operators must make agreements on fares, which results in higher responsibilities for the operators and less oversight and management of these agreements for TrIP. The operators loose flexibility, but gain in the sense that the risk for possible outages is lower and user satisfaction is higher.

DECISION 12: FARE MANAGEMENT

STATUS

Open

ARCHITECTURAL SUMMARY

In the context of	Storing and managing trip fares
Facing	Maintainability and Usability
To achieve	Quick fare calculation
We considered	TrIP calculating fares
	Transport operators calculating fares
And decided for	TrIP calculating fares
Because	It allows for control over revenue payout and limits third-party dependency
Accepting	Maintenance and updating of fare calculation processes for different transport operators

CONCERN

As a user, I want my trip costs to be calculated correctly, so that I do not pay too much for my trips.

As a transport operator, I want to be able to determine and update fare prices regularly, so that I have control over my financial health.

CONTEXT

Following Decision 10 and 11 regarding the fare revenue payout and fare tariffs, it must be decided how the fare management is settled. The options describe the operators managing and calculating the fare upon check-out or whether this is the responsibility of TrIP.

CRITERIA

- Maintainability
- Accessibility
- Usability

OPTION 1: TRIP CALCULATING FARES

This option results in TrIP managing the fares, meaning that TrIP is responsible for storing the fares for routes and operators. When a passenger checks out, the TrIP module retrieves the routes sections the passenger has travelled and the corresponding fare prices, meaning that calculating the fare is done by TriIP systems. Thus, this makes the architecture less dependent on third-party interfaces. However, the main drawback is that the transport operators are required to provide TrIP with the routes they are active on and the corresponding fares and also need to update these fares and routes in the TrIP portal when these are changed.

OPTION 2: TRANSPORT OPERATORS CALCULATING FARES

With this implementation, the transport operators are responsible for calculating and providing fares of passenger trips. This means that the TrIP system has to communicate with the transport operators every time a passenger checks out. This option creates a dependency on third-party interfaces, that may not be present yet in the transport operator systems. The main advantage is that updating fares and routes only has to happen in one place, avoiding inconsistencies.

DECISION

Criteria/Option	Option 1	Option 2
Maintainability	-	+
Usability	+	-
Reliability	+	-

We opted for option 1, since relying on a third-party interfaces increases waiting time on check-in, reducing usability. Reliability is also higher, since there is almost no communication necessary between the TrIP system and transport operators, with all fare and route data being stored within the TrIP system.

CONSEQUENCES

Transport operators will be responsible for updating fares and routes in the TrIP system, decreasing maintainability. The fare calculation processes might also be different between transport operators, and need to be implemented in the TrIP system.

DECISION 13: QUEUEING DURING NETWORK OUTAGE

STATUS

Open

ARCHITECTURAL SUMMARY

In the context of	Checking in and out at payment terminals
Facing	Availability, Usability and Security
To achieve	Correct fare calculation
We considered	Making travelling free
	Queueing requests
	Connectivity backup
And decided for	Queueing requests & Connectivity backup
Because	It has the highest possible save of revenue in case of a network outage
Accepting	Increased implementation and hardware costs & higher fraud risks

CONCERN

As a transport operator, I want to be always paid out fare revenues, so that my operations remain profitable.

As a user, I want to be able to always check in and out, so that I do not have missed check-ins or -outs resulting in incorrect fare calculation.

As a rogue user, I want to be able to make use of system outages, so that I can travel for free.

CONTEXT

In the event of a local or broader network outage, the payment terminals are unable to connect with the TrIP systems that handle the check-in and -out processes and calculate the fares accordingly. This has multiple consequences. When passengers are unable to check-in, but do check-out, the system register this as a maximum credit route, meaning that the passenger will probably get charged for an amount they did not realistically owe. When the passenger does check-in but is unable to check-out upon arrival due to an outage, the same situations occurs and full fare is charged. There are also passengers that can do neither, resulting in the operators receiving zero revenue for such passengers. Additionally, when conductors are unaware of very short, local network outages and passengers were unable to check-in but get checked by te conductor, this may result in inappropriate fines, difficult situations of judgment for the conductor or even violent actions against personnel.

CRITERIA

- Availability
- Usability
- Security

OPTION 1: MAKING TRAVELLING FREE

In this option, when a network outage occurs, the payment terminal is placed in out-of-service mode automatically, letting passengers know they cannot check-in or out. Passengers are not required to check-in and can travel for free. This results in high usability, and no security risks but is considerably damaging for system availability. This especially holds for regions that have frequent network outages or for the mobile payment terminals in busses for example. The latter payment terminals do not have a wired connection, which results in less stable connectivity.

Although most passengers will enjoy the free trip, it may result in difficult to judge situationships for conductors if these are unaware of the outages or unavailability of some payment terminals at a certain stop. If the passenger argues that he or she was not able to check-in, it is difficult for the conductor to verify this, making it possible for more passengers to pretend they could not check-in.

OPTION 2: QUEUEING REQUESTS

This option includes a local queueing process at the terminals. When a chipcard is recognized, the terminal attempts communication with the TrIP services. If this fails, it locally stores the chipcard instance using on-device non-persistent ephemeral storage. When connectivity is restored, the payment terminal will attempt to asynchronously send the requests of the earlier checked-in or -out chipcards and a header conveying information that it concerns an asynchronous check-in/-out.

The check-in and -out module determines whether these requests are check-ins or check-outs based on the timestamps and waits 24 hours for check-in and check-out requests to be matched and completed. If there is a check-in but the check-out request was never received, the trip is discarded and no amount is charged.

OPTION 3: CONNECTIVITY BACKUP

The stationary payment terminals are connected by ethernet cables. However, in case of local network issues, the payment terminals are equipped with mobile connectivity modules as well to switch to in case of inconnectivity. This network traffic is however pricier for the operators and results in higher operational costs. Also the prices of the payment terminals will increase due to the added hardware.

DECISION

Criteria/Option	Option 1	Option 2	Option 3
Availability	--	++	+
Usability	++	0	0
Security	0	-	++

The decision is made to choose for option 2 and 3. Both have a positive impact on the availability of the system and do not affect the usability. Option 2 does harm the security slightly, but the rsisk can be easily mitigated and the advantage of choosing for this option as well is high due to high availability.

CONSEQUENCES

The consequence of choosing for option 2 and 3 results in added availability of the system, making it virtually always possible for passengers to check-in and -out, whether this request is then actually processed or not. The usability remains mostly the same, as less users will be able to travel uncharged which might decrease usability. However, it offers clarity for passengers that they must always check-in and -out no matter the situation and does not confuse them with out-of-service modes of terminals. It will also avoid discussions with personnel.

The second option however does hurt the security of the system, as the chipcard data of unprocessed requests remains on the payment terminals until they have been processed. However, this can be mitigated with the chosen storage method where after the timetable day ends a new session is started and the previous container including memorized chipcards are discarded. Additionally, if someone tampers with the terminals and opens the terminal, power is disconnected and since it involves stateless storage, the chipcard information is lost.

DECISION 14: CHIPCARD STATE STORAGE

STATUS

Open

ARCHITECTURAL SUMMARY

In the context of	Checking in and out at payment terminals
Facing	Reliability, Availability
To achieve	Reduction of fraud and system faults
We considered	Server side state storage
	Chipcard state storage
And decided for	Server side state storage & Chipcard state storage
Because	Server side storage is needed for the system to function properly and chipcard storage is used as fault-tolerance
Accepting	Increased implementation complexity

CONCERN

As a transport operator, I want to minimize the events of chipcard fraud, so that I do not miss any revenue streams.

As a passenger, I want my information to be secure and not fall into the wrong hands, so that identity fraud or chipcard duplication is avoided.

As a transport operator, I want to be always paid out fare revenues, so that my operations remain profitable.

As a user, I want to be able to always check in and out, so that I do not have missed check-ins or -outs resulting in incorrect fare calculation.

As a rogue user, I want to be able to make use of system flaws, so that I can travel for free.

CONTEXT

With the made decisions for terminal queueing, the TrIP services may not have received a check-in request from a terminal in case of a network outage. However, when a conductor comes and checks the chipcard of the passenger, there is no checked-in flag while the passenger is checked-in, but the request is not yet communicated. This required for an addition to server side state storage only.

CRITERIA

- Reliability
- Availability

OPTION 1: SERVER SIDE STATE STORAGE

In this option, only the TrIP systems know the status of any given chipcard. This means that from the chipcard itself does not become clear whether a passenger is checked-in or not. This harms the system availability due to earlier choices.

OPTION 2: CHIPCARD STATE STORAGE

This option offers the feature where the check-in/check-out state of the passenger is stored in the chipcard. When a passenger checks-in at a payment terminal, the NFC chip is read first. This information includes the last flag (checked-in or -out) and some details such as which stations and timestamp that event occurred at. Then the NFC chip is written to with a new flag.

DECISION

Criteria/Option	Option 1	Option 2
Reliability	+	++
Availability	+	++

The decision is made to choose for both options, as these act complementary. The server-side state storage is leading in most cases and will be used to calculate fares and charge credit. However, in certain occasions, it is needed to check the passenger's action and travel history without the need or availability of the TrIP systems.

When a passenger checks-in and -out, the old flag is sent as well, letting the system know what the last state of the chipcard should be according to the chipcard information. If this information does not match, it can be tracked and if it occurs frequently, it may point to chipcard fraud or tampering. Choosing for both increases system reliability and availability.

CONSEQUENCES

The consequence of this choice is minimal. NFC chips are built to store small amounts of data such as state flags. NFC readers at the payment terminals will have to write some data as well. However, this interfacing comes standard on most hardware modules. Therefore, there is no need to perform any major infrastructural or hardware changes.

CONTEXT VIEWPOINT

VIEW: STAKEHOLDER MODEL

MODEL

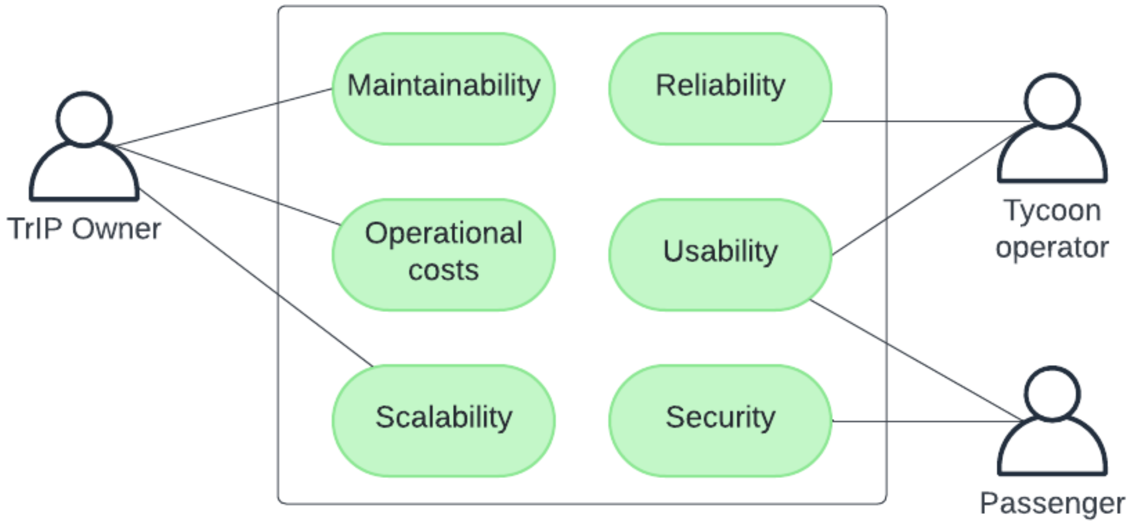

Figure 1

DESCRIPTION

Figure 1 presents the stakeholder model for the TrIP system. The main stakeholders include the TrIP system owner, the tycoons (3) and passengers. These combined have six main concerns, being 'maintainability', 'reliability', 'operational costs', 'usability', 'scalability' and 'security'.

GLOSSARY OF ELEMENTS

Id	Name	Description
1	TrIP Owner	The owner of TrIP is a stakeholder, since they have to develop and maintain the TrIP system.
2	Tycoon operator	The tycoon operators are stakeholders, since they rely on the TrIP system for profits.
3	Passenger	Passengers are stakeholders, since they will be using/interacting with the TrIP system.
4	Maintainability	For the TrIP owner, maintainability is an important quality for the system to have. The effort for aintaining and building the system should be minimal.
5	Reliability	For the tycoon operators, the reliability of the system is really important. Passengers should always be able to pay for their travels.
6	Operational costs	A second important quality that the TrIP owner wants the system to have is low operational costs.

7	Usability	Tycoon operators want to be able to easily add or remove payment terminals, so usability is an important quality for them. This quality is important for passengers as well, since they will be the ones actively using the system.
8	Scalability	For the TrIP owner, scalability of the system is a very important quality. New tycoons should be easy to connect to the system.
9	Security	For passengers, the security of the system is important. Passengers want their personal information to be safe from unauthorized access.

ANALYSIS ON PERSPECTIVES

Describe how the view satisfies the different perspectives.

VIEW: USER STORIES

MODEL

Passenger

As a passenger, I want to be able to commute from a certain station to another station by train, so that I can reach my desired destination.

As a passenger, I want to be able to pay for a single fare commute by train, so that I do not risk a fine.

As a passenger, I want to be able to have subscriptions for any of the three tycoons, so that I can travel more economically.

As a passenger, I want to be able to easily transfer between trains operated by different tycoons without multiple payments, so that my commute is seamless and cost-effective.

As a passenger, I want to receive a confirmation of my payment or subscription validation as soon as I check-in at a station, so that I am assured my journey is covered.

As a passenger, I want to be able to update or cancel my subscriptions through a user-friendly interface, so that I can manage my travel expenses more effectively.

As a passenger, I want to be able to report issues or get support easily if there's a problem with my fare or subscription, so that quick resolution and satisfaction is ensured.

As a passenger, I want my personal data to be shielded from unauthorized access, so that my privacy is protected and my trust in the TrIP system is maintained.

Tycoon operators

As a tycoon operator, I want to integrate my current subscription and fare system with the TrIP system, so that I can maintain my revenue streams without major changes to my infrastructure.

As a tycoon operator, I want to be able to access detailed reports on passenger usage and payment statistics, so that I can make informed decisions about my service offerings.

As a railway tycoon, I want the ability to add or remove payment terminals at stations, to adapt to changing passenger flows without significant downtime.

As a transport operator, I want to be able to close or open stations, to adapt to changing passenger travel needs.

As a transport operator, I want to ensure that all transactions are processed reliably, even during peak times, to maintain passenger satisfaction and trust.

As a transport operator, I want the flexibility to adjust fares and subscription options in response to market conditions, without extensive system reconfiguration.

As a bus tycoon, I want the ability to adjust the amount of payment terminals depending on bus capacity and crowdedness.

TrIP Owner

As the TrIP owner, I want to ensure that the system architecture supports scalability, so that it can handle growing numbers of passengers and transactions without degradation in performance.

As the TrIP owner, I want to implement robust security measures, to protect passenger data and prevent unauthorized access.

As the TrIP owner, I want to create a maintenance-friendly system, allowing for easy updates and bug fixes with minimal downtime.

As the TrIP owner I want to ensure the system integrates seamlessly with each tycoon's existing infrastructure, for smooth operation and user experience.

As the TrIP owner, I want to design an intuitive user interface for the payment terminals, to facilitate easy use by passengers of all ages and tech-savviness.

DESCRIPTION

Short description of the view

GLOSSARY OF ELEMENTS

Id	Name	Description

ANALYSIS ON PERSPECTIVES

Describe how the view satisfies the different perspectives.

VIEW: USE CASES AND INTERACTION SCENARIOS

MODEL

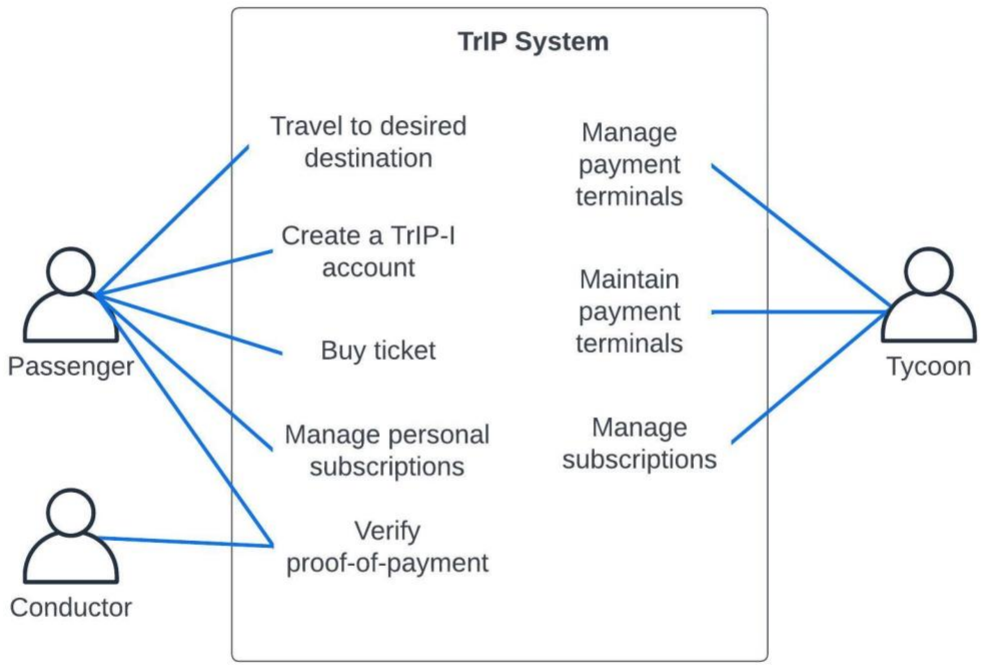

Figure 2

DESCRIPTION

Figure 2 is a use case model for the TrIP system. The model showcases the users of the system and cases about how the system is used.

GLOSSARY OF ELEMENTS

Id	Name	Description
1	Passenger	Main user of the system that would like to travel to a desired destination
2	Conductor	Verifies that the passengers have paid for their ride
3	Tycoon	Ensures that the payment terminals are managed and maintained and manages the travel subscriptions
4	TrIP System	System that is used by users with various use cases

USE CASES

Use Case component	Description
Name	Travel to desired destination
Goal	Passenger successfully takes a form of public transport in order to arrive at the desired destination
Actors	Passenger

Primary actor	Passenger
Trigger	Passenger wants to travel to another location by using public transport
Precondition	Passenger must have a valid ticket or TripCard
Main success scenario	1. Passenger arrives at train station 2. Passenger selects mode of transport 3. Passenger checks in at payment terminal of selected mode of transport 4. Passenger takes selected mode of transport to desired destination 5. Train passenger checks out at the payment terminal of the desired destination
Alternative paths	1. Train passenger does not check in at payment terminal 2. Payment terminals are broken

---

Use Case component	Description
Name	Create TrIP-I account
Goal	The passenger successfully creates a TrIP-I account in order to request a Personalized TripCard
Actors	Passenger, TrIP System
Primary actor	Passenger
Trigger	Passenger wants to request a Personalized TripCard
Precondition	-
Main success scenario	1. Passenger goes online in order to create a TrIP-I account 2. Passenger provides necessary personal information 3. Personal information is verified by the TrIP System 4. Passenger requests a TripCard
Alternative paths	1. Personal information is not verified by the TrIP System

Use Case component	Description
Name	Buy ticket
Goal	Passenger successfully buys train ticket for desired train ride(s)
Actors	Passenger
Primary actor	Passenger
Trigger	Passenger wants to buy a ticket in order to travel to another location by public transport without the use of a TripCard
Precondition	Passenger must not use/have a valid TripCard

Main success scenario	1. Passenger goes online to purchase a ticket 2. Passenger selects the day of travel 3. Passenger selects station or bus stop where they start their ride 4. Passenger selects desired destination for their ride 5. Passenger selects the format of the ticket 6. Passenger pays for the ticket 7. Passenger receives the ticket
Alternative paths	1. Due to internet outage, the passenger is unable to go online to buy a ticket 2. Passenger selects wrong day of travel 3. Passenger selects wrong ticket format 4. Passenger does not pay for the ticket 5. Ticket is not received

Use Case component	Description
Name	Verify proof-of-payment
Goal	The Conductor verifies if everyone onboard the train has a valid TripCard or ticket
Actors	Passenger, Conductor
Primary actor	Conductor
Trigger	Mode of transport leaves for another station or stop
Precondition	Mode of transport must have a conductor and passengers onboard
Main success scenario	1. Mode of transport departs 2. Conductor checks each passenger’s TripCard or ticket 3. Each passenger's TripCard or ticket is verified
Alternative paths	1. Passenger does not have a valid TripCard or ticket 2. Passenger has not checked in with their TripCard or ticket

Use Case component	Description

Name	Manage personal subscriptions
Goal	Passengers are successfully able to change their subscriptions offered by the tycoons
Actors	Passenger
Primary actor	Passenger
Trigger	Passenger wants to change something about the subscriptions they currently have
Precondition	Passenger has a TrIP-I account
Main success scenario	1. Passenger logs into TrIP-I account 2. Passenger selects subscriptions they want and removes subscriptions he does not want 3. TrIP-I account is updated to include other subscriptions
Alternative paths	1. Passenger fails to log into TrIP-I account 2. Passenger selects wrong subscriptions to add or remove 3. TrIP-I account fails to include new update on subscriptions

Use Case component	Description
Name	Manage payment terminals
Goal	Tycoons are able to decide how many terminals each bus or station needs and can remove or add payment terminals if needed
Actors	Tycoon
Primary actor	Tycoon
Trigger	Tycoon wants to remove or add a payment terminal somewhere
Precondition	Payment terminal must be placed at a valid location (station or inside a bus)
Main success scenario	1. Tycoon selects location where terminal is added or removed 2. Tycoon installs new payment terminal or removes old terminal 3. Tycoon notifies TrIP of addition or removal of terminal
Alternative paths	1. Wrong terminal is removed 2. TrIP is not notified of terminal being removed or added

Use Case component	Description

Name	Maintain payment terminals
Goal	Tycoon ensures that the payment terminals are working and function correctly so that all passengers are able to check in and out when needed
Actors	Tycoon
Primary actor	Tycoon
Trigger	Tycoon is alerted that a payment terminal is not functioning correctly
Precondition	Tycoon has maintenance worker available
Main success scenario	1. Tycoon elects maintenance worker to check why payment terminal is not functioning correctly 2. Maintenance worker goes over to payment terminal that is not functioning correctly 3. Maintenance worker fixes the problem of the payment terminal 4. The payment terminals all work correctly
Alternative paths	1. Payment terminal cannot be fixed by maintenance worker 2. Payment terminal is functioning correctly, but falsely sent an alert

Use Case component	**Description**
Name	Manage subscriptions
Goal	As a tycoon operator, I want to add, remove or adjust my subscriptions based on passenger needs
Actors	Tycoon, passengers
Primary actor	Tycoon
Trigger	Tycoon wants to change something about their subscriptions
Precondition	Passenger must have TrIP-I account
Main success scenario	1. Tycoon sends subscription change request 2. Change request is accepted 3. Subscriptions change

	4. Passengers with TrIP-I accounts are notified that the subscriptions have changed
Alternative paths	1. Change request is denied by subscription management system 2. Passengers with TrIP-I accounts have not received notification about the subscription change

ANALYSIS ON PERSPECTIVES

Describe how the view satisfies the different perspectives.

VIEW: TICKETING PROCESS MODEL

MODEL

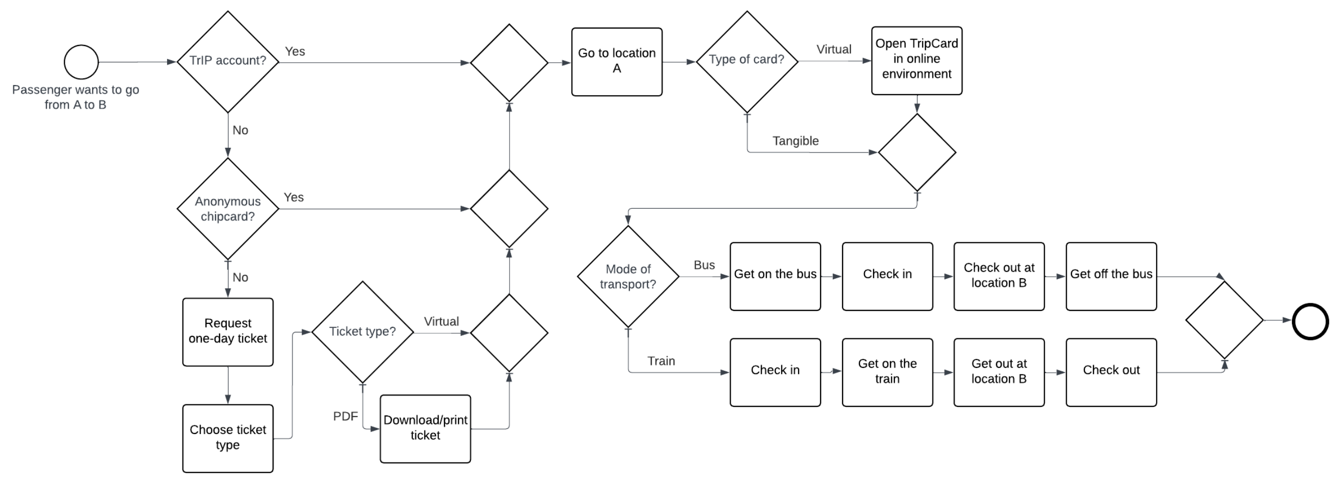

Figure 3

DESCRIPTION

Figure 3 presents a process model clarifying the context for ticketing. When a passenger wants to travel from a location A to B, there are multiple ways to pay for the trip, and therefore multiple sets of steps a passenger can take to get on the train/bus with proof-of-payment.

GLOSSARY OF ELEMENTS

Id	Name	Description
1	TrIP account?	The first step is checking whether the passenger has a TrIP account. If they do, no further preparation is needed.
2	Anonymous TripCard?	If the passenger does not have a TrIP account, we need to check whether they have an anonymous TripCard. If they do, no further preparation is needed.
3	Request one-day ticket	If the passenger does not have either a TrIP account or an anonymous TripCard, they should request a one-day ticket for their train.

4	Choose ticket type	After requesting a one-day ticket, the passenger should choose how they want to receive the ticket, either virtual or in pdf.
5	Ticket type?	If the passenger has requested a one-day ticket and chosen a ticket type, we need to check what choice they made.
6	Download/print ticket	If the passenger has requested a one-day ticket in pdf, they need to download and optionally print the ticket.
7	Go to location A	When the passenger has prepared a way to check in, they can go to the location they want to travel from.
8	Type of card?	Check in cards can either be virtual or tangible.
9	Open TripCard in online environment	If the passenger has a virtual card, they should open it in the online environment now.
10	Mode of transport?	For the last steps, we have to differentiate between travelling by train or bus.
11/12	Get on the bus -> Check in	For bus passengers, payment terminals are on the bus, so they should get on the bus before checking in.
13/14	Check in -> Get on the train	For train passengers, payment terminals are in the station, so they should check in before getting on the train.
15/16	Check out at location B -> Get of the bus	Again, since the payment terminals are on the bus, passengers should first check out at their destination and then get off the bus.
17/18	Get of the train at location B -> Check out	Again, since the payment terminals are in the stations, passengers should first get off the train and then check out at the station.

ANALYSIS ON PERSPECTIVES

Describe how the view satisfies the different perspectives.

FUNCTIONAL VIEWPOINT

VIEW: FUNCTIONAL ARCHITECTURE MODEL

MODEL

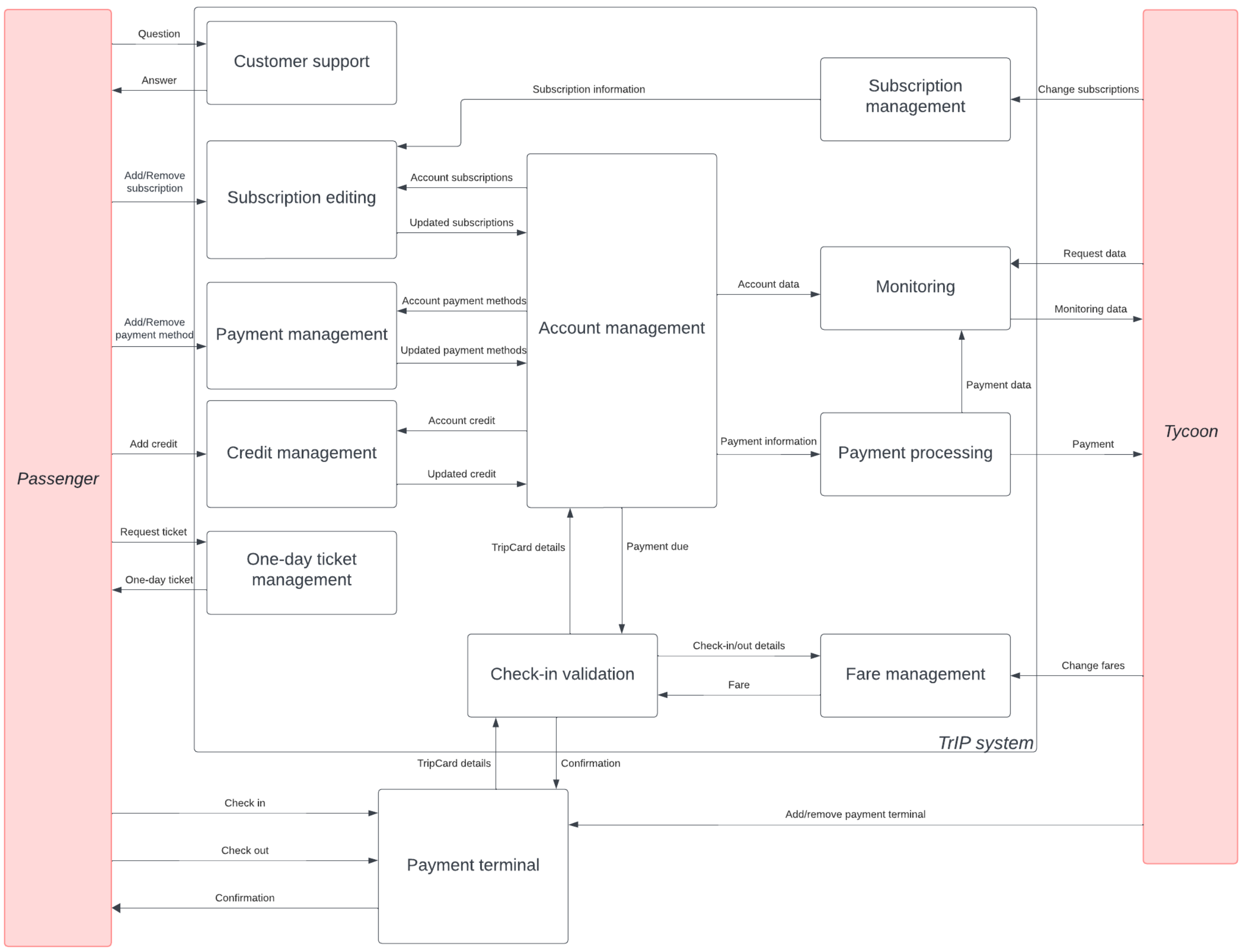

Figure 4

DESCRIPTION

Figure 4 presents the functional architecture model for the TrIP system. This model gives a top-level overview of the main functionalities that the new system will have, by visualizing the system's modules and interaction flows.

GLOSSARY OF ELEMENTS

Id	Name	Description
1	Passenger	The actor representing passengers that interacts with system modules as well as the Payment terminal actor.
2	Tycoon	The actor representing the tycoon operators that interacts with system modules as well as the Payment terminal actor.

3	Payment terminal	The actor representing the payment terminals at the stations and in busses, where passengers can check in/out with their TripCard. The payment terminal gives a confirmation of correct check in/out as well as the due payment by requesting information from the TRiP-system's check-in validation module. The payment terminals are managed by tycoons, meaning they can be added and removed from stations/busses by the tycoons.
4	TrIP system	The TrIP system containing all of the functional modules and interaction flows.
5	Account management	This module handles account information of passengers, including TripCard details, linked subscriptions, added payment methods and current credit. This data is requested by and sent to other modules within the TrIP-system.
6	Subscription management	This module handles all the different subscriptions that are defined by the tycoons. Information on specific subscriptions can be requested by the subscription editing module, and tycoons can directly interact with this module to 'change' (create, edit or remove) subscriptions.
7	Payment processing	This module handles every occurrence where a payment should be made. This can be periodically, based on subscriptions, or after passengers without a subscription have checked out. The payment information is provided by the account management or check-in validation module respectively.
8	Check-in validation	This module handles every occurrence where a passenger checks in/out at a payment terminal. It receives TripCard details from the payment terminal and checks whether the trip is covered by a subscription linked to that card. If not, payment information is sent to the payment processing module.
9	Fare management	This module handles the fares determined by the tycoons. Tycoons can directly interact with this module to change the standard fares between specific stations/stops. When a passenger has checked in and out, the Check-in validation module will request the standard fare for that trip from this module.
10	Monitoring	This module gathers monitoring data from both accounts and payments to create statistics. These statistics can be requested by the tycoons to provide insights on rush hours, station and bus stop popularity, route efficiency and subscription use.
11	Customer support	This module deals with questions from passengers and attempts to answers them.
12	Subscription editing	This module allows passengers with a TrIP-account to manage their subscriptions. Passengers are able to add and remove subscriptions to/from their account, and changes are sent to the account management module.

13	Payment management	This module allows passengers to manage the payment methods for their account. The payment method options include credit and direct bank transfer (SEPA). Changes are sent to the account management module.
14	Credit management	This module allows passengers to add credit to their account. The current credit for each account is managed by the account management module.
15	One-day ticket management	This module allows passengers without an account to buy one-day tickets for a specific trip. After payment, the ticket is sent to the passenger.

ANALYSIS ON PERSPECTIVES

Describe how the view satisfies the different perspectives.

INFORMATION VIEWPOINT

VIEW: ER DIAGRAM

MODEL

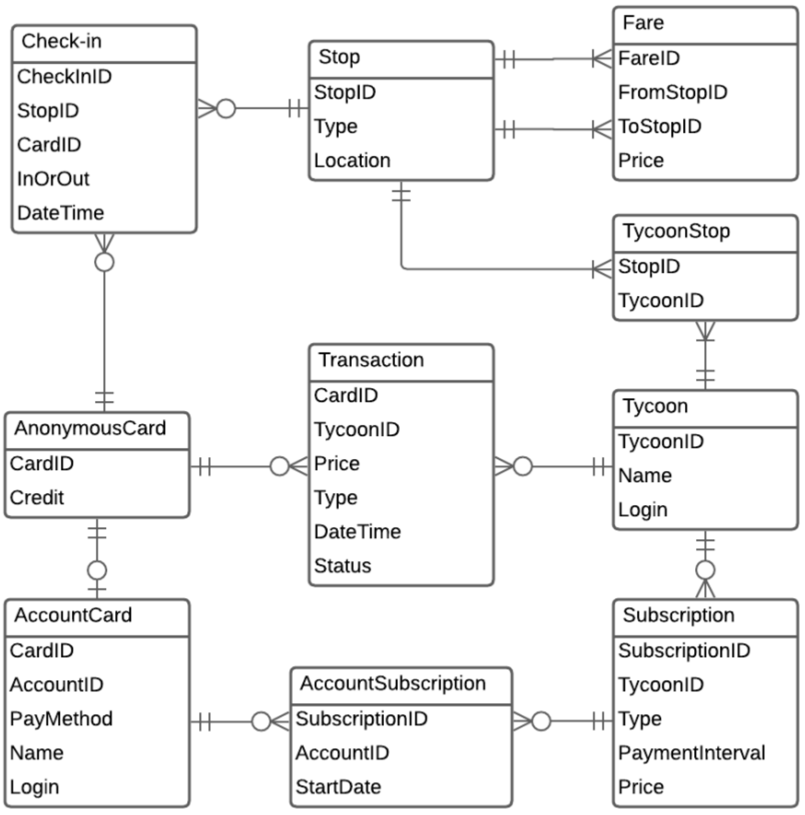

Figure 5

DESCRIPTION

Figure 5 presents the Entity-Relationship diagram (ER diagram) for the TrIP system. The model describes the data objects and their relationships, and can be implemented to define a relational database.

GLOSSARY OF ELEMENTS

Id	Name	Description
1	AnonymousCard	For every virtual or tangible TrIP-card, there is a matching *AnonymousCard* stored in the TrIP system. This entity holds the *cardID* and all information that can be linked to an anonymous TrIP-card, most importantly *credit*. An *AnonymousCard* can be linked to zero or one *AccountCards*, representing inheritance.
2	AccountCard	The *AccountCard* entity is always linked to exactly one *AnonymousCard*, enforcing inheritance. On top of the data stored in the *AnonymousCard* entity, the *AccountCard* stores account data like a passenger's

		name, *login* and active *payMethod*, and allows linking *Subscriptions* to the account.
3	Tycoon	Each of the tycoons involved with the TrIP system are represented as *Tycoon* entities in the database. To make sure tycoons can access the system services, a *login* can be stored for them. *Tycoons* are also linked to both *Stops* and *Subscriptions*, since they both are provided by the different tycoons.
4	Subscription	Each *Subscription* is linked to exactly one *Tycoon* and holds the information that defines a subscription that a passenger can buy. Each *Subscription* has a defined *price* per *paymentInterval* that passengers need to pay for, as well as a *type* defining when passengers get a discount on their trip and how high that discount is.
5	Account-Subscription	The *AccountSubscription* entity represents a *Subscription* bought by as passenger, and therefore linked to a *AccountCard*. Each *AccountCard* can have multiple active *Subscriptions* and each *Subscription* can be bought by many different passengers, which is represented in the model by one-to-many relationships with both. The *AccountSubscription* entity also holds the *startDate* attribute, that stores the date on which a subscription was bought. This is used to determine when the next payment should happen. There is a constraint in place on the data here, each *AccountCard* can be linked to a each *Subscription* at most once, meaning the *accountID-subscriptionID* pair is always unique.
6	Stop	For each station/bus stop that is governed by one of the *Tycoons* involved with the TrIP system, there is a *Stop* in the database. For each *Stop*, a *location* and *type* (bus or train) is stored.
7	TycoonStop	The *TycoonStop* entity is used to link *Stops* to *Tycoons*. This entity does not have its own attributes, but is only used to enforce the many-to-many relationship between *Tycoons* and *Stops*. This relationship is necessary since some *Stops* are managed by multiple tycoons.
8	Fare	To enable the calculation of travelling costs, *Fares* are stored in the TrIP system database. *Fares* are linked to exactly two *Stops* that are directly adjacent in any train/bus route, and store the standard *price* for travelling between these two *Stops*.
9	Check-In	The *Check-in* entity represents any occurrence where a passenger checks in or out at a payment terminal. It is linked to exactly one *AnonymousCard* and *Stop*, and stores the *dateTime* at which the check-in/out occurred as well as a Boolean stating whether the passenger checked in or out.
10	Transaction	*Transactions* are stored in the TrIP system database every time there is a payment from a passenger to a *Tycoon*. This can be either after a passenger travelled between two *Stops* managed by a *Tycoon* or when they

		have to pay for a bought *Subscription* (represented by the *type* attribute). Each transaction is linked to exactly one *AnonymousCard* and *Tycoon*, and stores the *price* paid by a passenger, the *dateTime* on which the payment took place and the *status* of the transaction (has the *Tycoon* received the payment?).

ANALYSIS ON PERSPECTIVES

Describe how the view satisfies the different perspectives.

VIEW: PASSENGER-SYSTEM DATAFLOW

MODEL

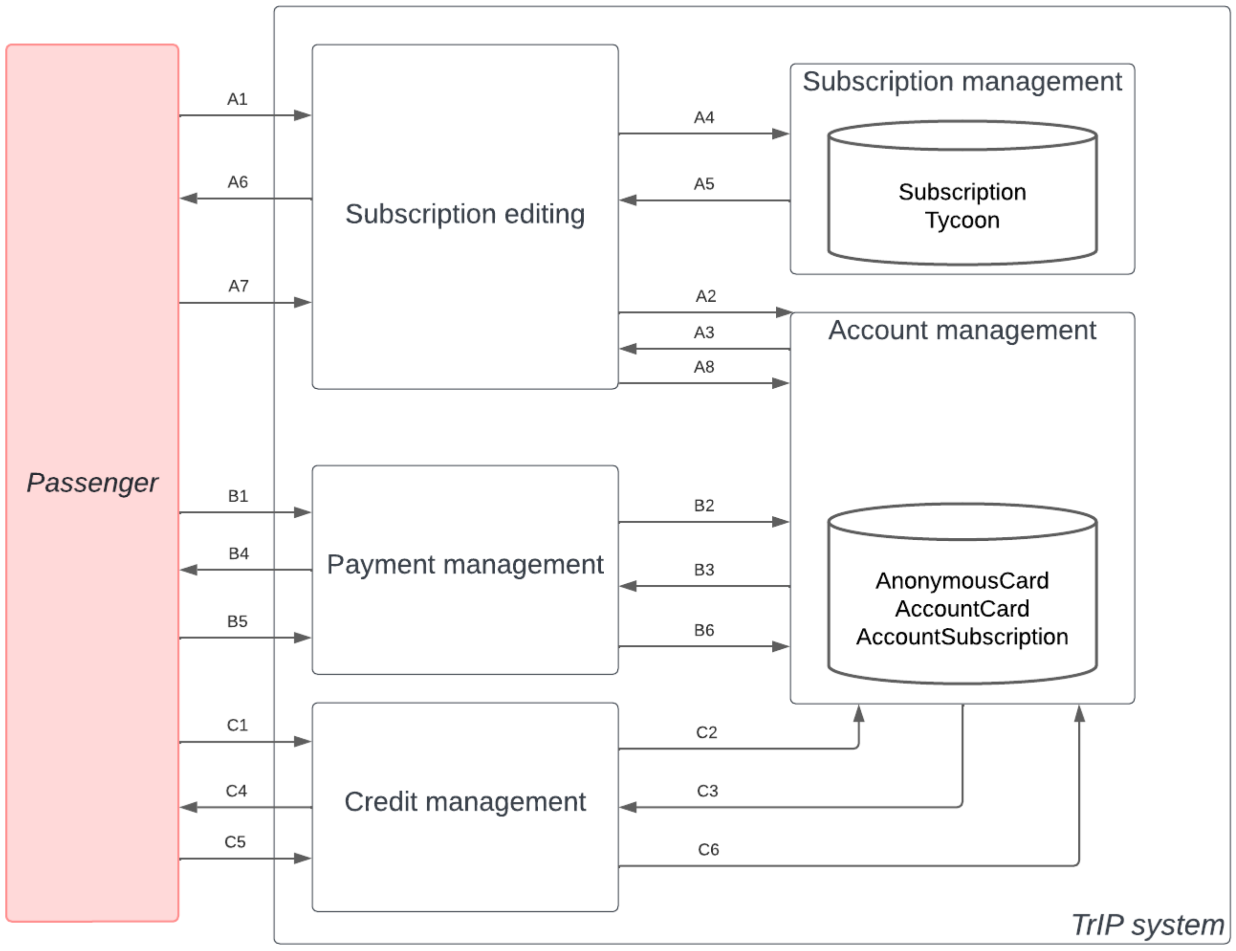

Figure 6

DESCRIPTION

Figure 6 presents the dataflow model for passenger-system interaction. The model uses the same system modules as defined in the functional architecture model (Figure 4). Passengers have three main ways to directly interact with the system: Editing subscriptions (flow A), editing payment methods (flow B) and adding credit (flow C). This model describes the exact information that is sent between the different system modules and passengers, as well as the order in which data is sent. For simplicity, the arrows representing data being sent are labelled with only the corresponding flow and their ordering number in the model. See the glossary of element below for details on the arrows.

GLOSSARY OF ELEMENTS

Only the system modules that store data are included in the table below, since all modules are already covered in the glossary of elements for the functional architecture model (Figure 4). The name column shows the data entities and/or their attributes that are being sent, as defined in the ER diagram (Figure 5). Entities are listed with an uppercase letter (e.g. Subscription), attributes with a lowercase letter (e.g. login) and other messages are listed in italics (e.g. *Confirmation*).

Id	Name	Description
	Subscription management	The Subscription management module holds a data storage for all of the Subscriptions and Tycoons that are known by the TrIP system. This data can be requested by the Account management module.
	Account management	The Account management module holds all data that has to do with passengers: AnonymousCard, AccountCard and AccountSubscription. This data can be requested and updated by the other system modules in the model.
A1	login, cardID	Dataflow A represents the subscription editing service. The first step in this process is authorization, so the Passenger has to send their login and cardID to the Subscription editing module.
A2	login, cardID	The Subscription editing module forwards the authorization data to the Account management module, which can confirm whether the cardID-login pair matches with any known TrIP-card.
A3	*Confirmation*, Account-Subscription	After the Account management module has determined whether the authorization was valid, it sends a *Confirmation* message as well as all the AccountSubscriptions linked to the given cardID back to the Subscription editing module. If the authorization could not be validated, only the *Confirmation* message will be sent.
A4	*Request*	Step A4 and A5 happen in parallel with A2 and A3, the Subscription editing module sends a *Request* to the Subscription management module to send the currently available Subscriptions.
A5	Subscription	After receiving a *Request* for Subscription data, the Subscription management sends back all available Subscriptions.
A6	*Confirmation*, Account-Subscription, Subscription	After receiving a *Confirmation* message from the Account management module as well as the Subscription data from the Subscription management module, all the data is sent to the Passenger. The Passenger can now see/edit their active

		AccountSubscriptions and add new Subscriptions to their account. If the *Confirmation* message was negative, meaning authorization could not be validated, only the *Confirmation* message will be sent here.
A7	Account-Subsciption	Step A7 and A8 can only happen when authorization was validated and only after step A6. If the Passenger has made any changes/additions to their AccountSubscriptions, the updated list is sent to the Subscription editing module.
A8	Account-Subscription	After receiving an updated list of AccountSubscriptions from a Passenger, the list is forwarded to the Account management module. The data is then used to update the data storage there.
B1	login, cardID	Dataflow B represents the payment method editing service. The first step in this process is authorization and is very similar to A1, with the data being sent to the Payment management module here.
B2	login, cardID	Similar to A2, the Payment management module forwards the authorization data to the Account management module..
B3	*Confirmation*, payMethod	After authorization is validated, a *Confirmation* message is sent back to the Payment management module, alongside the active payMethod linked to the given cardID. If the authorization could not be validated, only the *Confirmation* message is sent here.
B4	*Confirmation*, payMethod	The Payment management module forwards the *Confirmation* message and payMethod (if the authorization was validated) to the Passenger.
B5	payMethod	Step B5 and B6 can only happen when authorization was validated and only after step B4. If the Passenger changes their active payMethod, this update is sent to the Payment management module.
B6	payMethod	The Payment management module forwards the changed payMethod to the Account management module. This attribute is then updated in the data storage.
C1-6	login, cardID	Dataflow C represents the credit management service. This process is very similar to dataflow B, with communication going though the Credit management module instead of Payment management, and the credit attribute being sent/updated instead of the payMethod.

ANALYSIS ON PERSPECTIVES

Describe how the view satisfies the different perspectives.

VIEW: PASSENGER-TERMINAL-SYSTEM DATAFLOW

MODEL

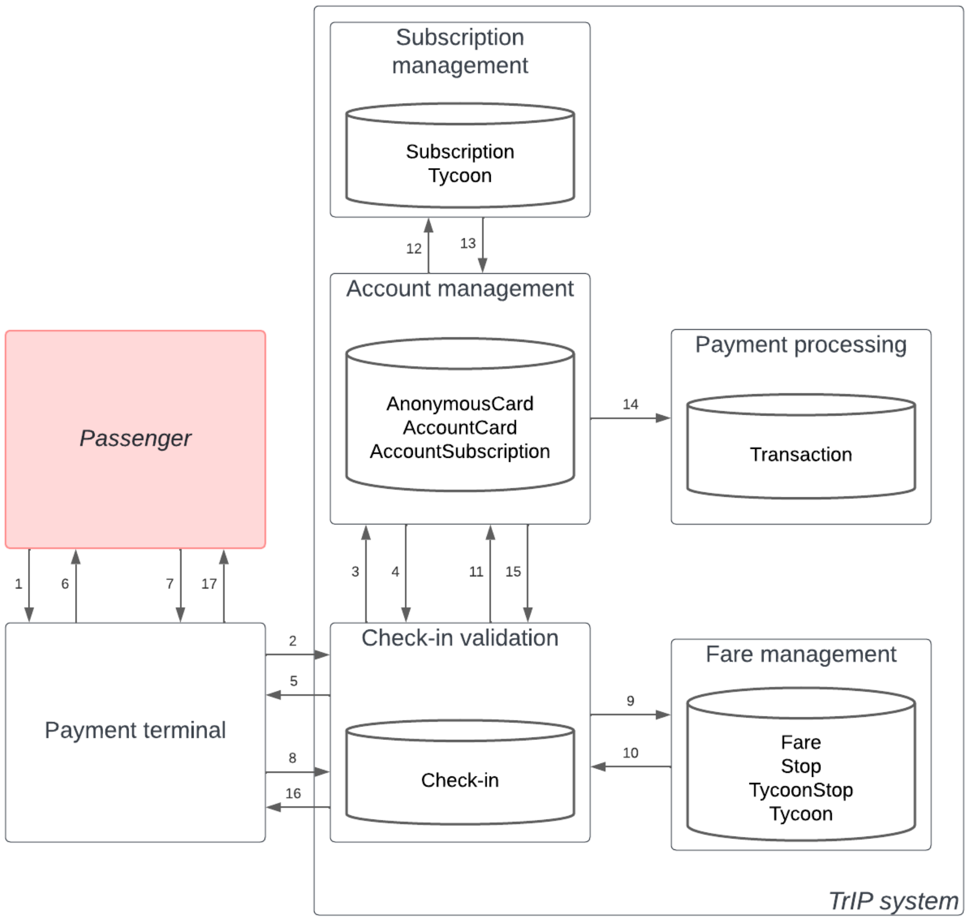

Figure 7

DESCRIPTION

Figure 7 presents the dataflow model for passenger-terminal-system interaction. The model uses the same system modules as defined in the functional architecture model (Figure 4). Passengers can either check in or check out at a payment terminal, which sets the data flow in motion. This model describes the exact information that is sent between the different system modules and passengers, as well as the order in which data is sent. For simplicity, the arrows representing data being sent are labelled with only their ordering number in the model. See the glossary of element below for details on the arrows.

GLOSSARY OF ELEMENTS

The name column shows the data entities and/or their attributes that are being sent, as defined in the ER diagram (Figure 5). Entities are listed with an uppercase letter (e.g. Subscription), attributes with a lowercase letter (e.g. login) and other messages are listed in italics (e.g. *Confirmation*).

Id	Name	Description
	Check-in validation	The Check-in validation module stores the history of Check-ins. Each time a Passenger checks in or out, a new Check-in instance is created.

	Fare management	The Fare management module stores all of the Fares defined by the tycoons. Specific Fares can be requested by the Check-in validation module.
	Account management	The Account management module holds all data that has to do with passengers: AnonymousCard, AccountCard and AccountSubscription. This data can be requested and updated by the other system modules in the model.
	Subscription management	The Subscription management module holds a data storage for all of the Subscriptions and Tycoons that are known by the TrIP system. This data can be requested by the Account management module.
	Payment processing	The Payment processing stores the history of Transactions between Passenger and Tycoon. Transactions are always provided by the Account management module.
1	cardID	The dataflow in this model is started when a Passenger checks in at a Payment terminal. The cardID is read from the Passenger's TrIP-card by the Payment terminal.
2	cardID, Stop	After reading a cardID, the Payment terminal forwards the cardID to the Check-in validation module, along with the Stop linked to the Payment terminal
3	Check-in	After receiving a cardID and Stop, the Check-in validation module determines whether the passenger has checked in or out, based on previously stored Check-ins with the same cardID. In case of a check-in, this step is performed and in case of a check-out, step 9 is performed. In step 3, a Check-in instance is created, stored an sent to the Account management module.
4	*Confirmation*, credit	After receiving a Check-in instance with type check-in, the Account management service retrieves the credit associated with the same cardID. If the cardID is known, the credit and a *Confirmation* message are sent to the Check-in validation module. If the cardID is not known, only the *Confirmation* message is sent.
5	*Confirmation*, credit	The *Confirmation* message and credit are forwarded to the Payment terminal.
6	*Confirmation*, credit	If check-in was successful, the Payment terminal will display the *Confirmation* as well as the current credit to the Passenger. If the check-in was not successful, an error message will be displayed.
7	cardID	This step is identical to step 1, but represents the Passenger checking out instead.
8	cardID, Stop	This step is identical to step 2, but represents the Passenger checking out instead.
9	Check-in	After receiving a cardID and Stop, the Check-in validation module determines whether the passenger has checked in or out, based on previously stored Check-ins with the same cardID. In case of a check-in,

		step 3 is performed and in case of a check-out, step 9 is performed. In step 9, a Check-in is created and stored in the Check-in validation module. This new Check-in along with the most recent Check-in instance that was stored are sent to the Fare management module.
10	Check-in, *Price*, Tycoon	After receiving two Check-in instances, the Fare management module calculates the standard *Price* between the two Stops linked to the Check-ins. The Check-ins, calculated *Price* and the Tycoons managing the Stops are sent to the Check-in validation module.
11	Check-in, *Price*, Tycoon	The Check-in validation module then forwards both Check-ins, the calculated *Price* and the associated Tycoons to the Account management module.
12	Account-Subscription	The Account management module retrieves the AccountSubscriptions linked to the same cardID as the given Check-ins from storage, and sends them to the Subscription management module to request the actual Subscriptions. If the Check-ins are linked to an anonymous card or a AccountCard without Subscriptions, step 12 and 13 are skipped.
13	Subscription	After receiving AccountSubscriptions, the Subscription management module retrieves the linked Subscriptions from storage and sends them back to the Account management module.
14	Transaction	After receiving the Check-ins, calculated standard travelling *Price*, associated Tycoons and (optionally) active Subscriptions, the final price of the Passenger's trip is calculated and a Transaction instance is created. This Transaction is then sent to the Payment Processing module.
15	*Confirmation*, credit	After a Transaction instance has been made, credit is updated based on the price and the active payMethod of the Passenger's TrIP-card. A *Confirmation* and the updated credit are then sent to the Check-in validation module.
16	*Confirmation*, credit	The *Confirmation* message and updated credit are then forwarded to the Payment terminal.
17	*Confirmation*, credit	Lastly, the Payment terminal will display the *Confirmation* and the updated credit to the Passenger.

ANALYSIS ON PERSPECTIVES

Describe how the view satisfies the different perspectives

.

VIEW: TYCOON-SYSTEM DATAFLOW

MODEL

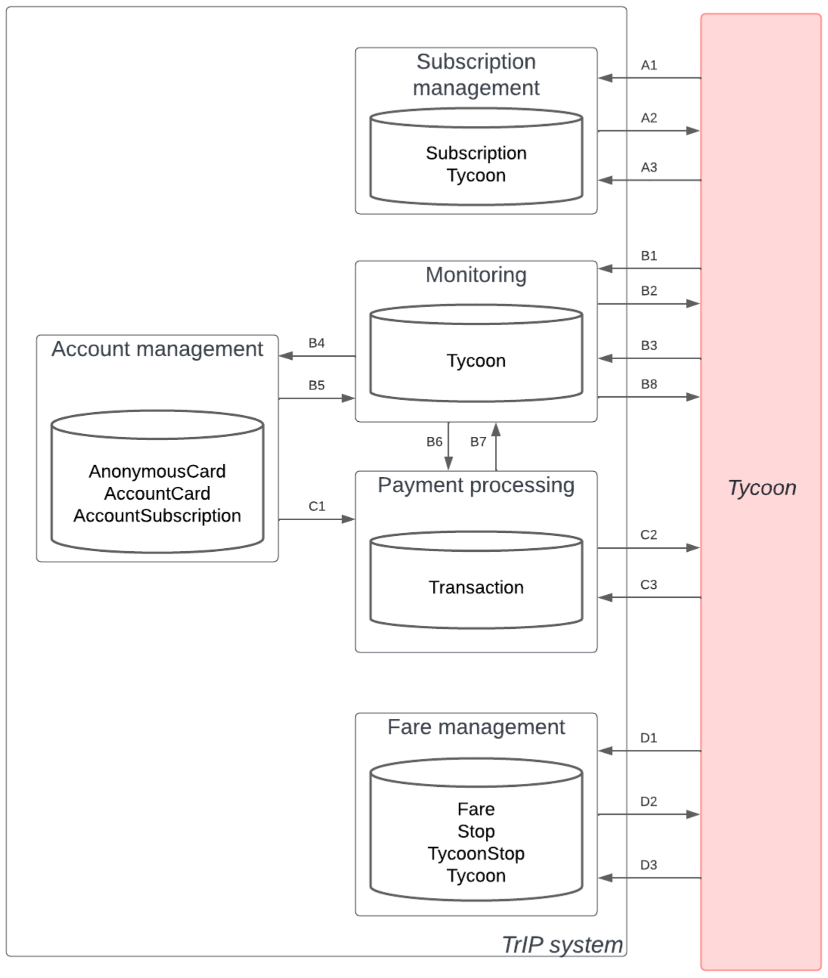

Figure 8

DESCRIPTION

Figure 6 presents the dataflow model for tycoon-system interaction. The model uses the same system modules as defined in the functional architecture model (Figure 4). There are four different interactions that can happen between a tycoon and the system: Editing subscriptions (flow A), requesting monitored data (flow B), payment status update (flow C) and editing fares (flow D). This model describes the exact information that is sent between the different system modules and tycoons, as well as the order in which data is sent. For simplicity, the arrows representing data being sent are labelled with only the corresponding flow and their ordering number in the model. See the glossary of element below for details on the arrows.

GLOSSARY OF ELEMENTS

Id	Name	Description
	Account management	The Account management module holds all data that has to do with passengers: AnonymousCard, AccountCard and AccountSubscription. This data can be requested by the Monitoring module.
	Subscription management	The Subscription management module holds a data storage for all of the Subscriptions and Tycoons that are

		known by the TrIP system. This data can be requested by the Account management module or changed by the Tycoons.
	Monitoring	The Monitoring module holds data about the Tycoons, allowing them to log in and access this module.
	Payment processing	The Payment processing stores the history of Transactions between Passenger and Tycoon. Transactions are always provided by the Account management module.
	Fare management	The Fare management module stores all of the Fares and Stops defined by the Tycoons, as well as Tycoon data. Tycoons can access this module to change Fares and Stops.
A1	tycoonID, login	Dataflow A represents the subscription management service that Tycoons can access. The first step of this process is authorization, so the Tycoon has to send a tycoonID and login to the Subscription management module.
A2	*Confirmation,* Subscription	After receiving a tycoonID and login, the Subscription management module validates the authorization and sends the current Subscriptions linked to the Tycoon as well as a *Confirmation* message back to the Tycoon. If authorization fails, only a *Confirmation* message is sent.
A3	Subscription	After receiving a *Confirmation* message and the current Subscriptions, the Tycoon can edit these Subscriptions or add new Subscriptions. The updated list is then sent back to the Subscription management module, where it is used to update the storage.
B1	tycoonID, login	Dataflow B represents the data monitoring service. The first step of this process is authorization, so the Tycoon needs to send the tycoonID and login to the Monitoring module.
B2	*Confirmation*	After receiving a tycoonID and login, the Monitoring module validates the authorization and sends a *Confirmation* message back to the Tycoon. If authorization fails, steps B3-B8 will not happen.
B3	*Data request*	After receiving a *Confirmation* message, the Tycoon can request data about their passengers, Subscriptions and Stops. This *Data request* is sent to the Monitoring module.
B4	*Data request*	Steps B4 and B5 can be performed in parallel with steps B6 and B7. In step B4, the *Data request* is forwarded to the Account management module.
B5	Account-Subscription	After receiving a *Data request*, the Account management module sends back the requested Subscription data, which will be based on the stored AccountSubscriptions linked to the Tycoon.
B6	*Data request*	Steps B4 and B5 can be performed in parallel with steps B6 and B7. In step B4, the *Data request* is forwarded to the Payment Processing module.

B7	Transaction, Stop	After receiving a *Data request*, the Payment processing module sends back the requested Transaction and Stop data, which will be based on the stored Transactions and Stops linked to the Tycoon.
B8	*Anonymized* Account-subscription, *Anonymized* Transaction	After receiving all data that the Tycoon requested from the Account management and/or Payment processing module, the data is anonymized and sent to the Tycoon.
C1	Transaction	Dataflow C represents the Transaction status update process. This process starts when the Account management module sends a new Transaction to the Payment processing module. This Transaction can either be a payment for an active Subscription or a payment for a passenger's trip. In case of the latter option, this step is equal to step 14 in the Passenger-Terminal-System dataflow model.
C2	*Transaction status request*	After some interval since the Transaction was received by the Payment processing module, or since the last status request, a *Transaction status request* is sent to the Tycoon that is linked to the Transaction.
C3	status	After receiving a *Transaction status request*, the Tycoon has to send back the status of the Transaction. The status of the Transaction is then updated in storage. If the Tycoon has received payment for the Transaction, no more *Transaction status requests* will be sent for it.
D1	tycoonID, login	Dataflow D represents the fare and stop management service that Tycoons can access. The first step of this process is authorization, so the Tycoon has to send a tycoonID and login to the Fare management module.
D2	*Confirmation*, Fare, Stop	After receiving a tycoonID and login, the Fare management module validates the authorization and sends the current Fares and Stops linked to the Tycoon, as well as a *Confirmation* message back to the Tycoon. If authorization fails, only a *Confirmation message* is sent.
D3	Fare, Stop	After receiving a *Confirmation* message and the current Fares and Stops, the Tycoon can edit them or add new Fares and/or Stops. The updated lists are then sent back to the Fare management module, where they are used to update the storage.

ANALYSIS ON PERSPECTIVES

Describe how the view satisfies the different perspectives.

CONCURRENCY VIEWPOINT

State model (LTS)
Protocol model --> sequence model, choreography model
Petri net maybe

DEVELOPMENT VIEWPOINT

Module structure model

UML

Protocol overview

DEPLOYMENT VIEWPOINT

VIEW: DEPLOYMENT MODEL

MODEL

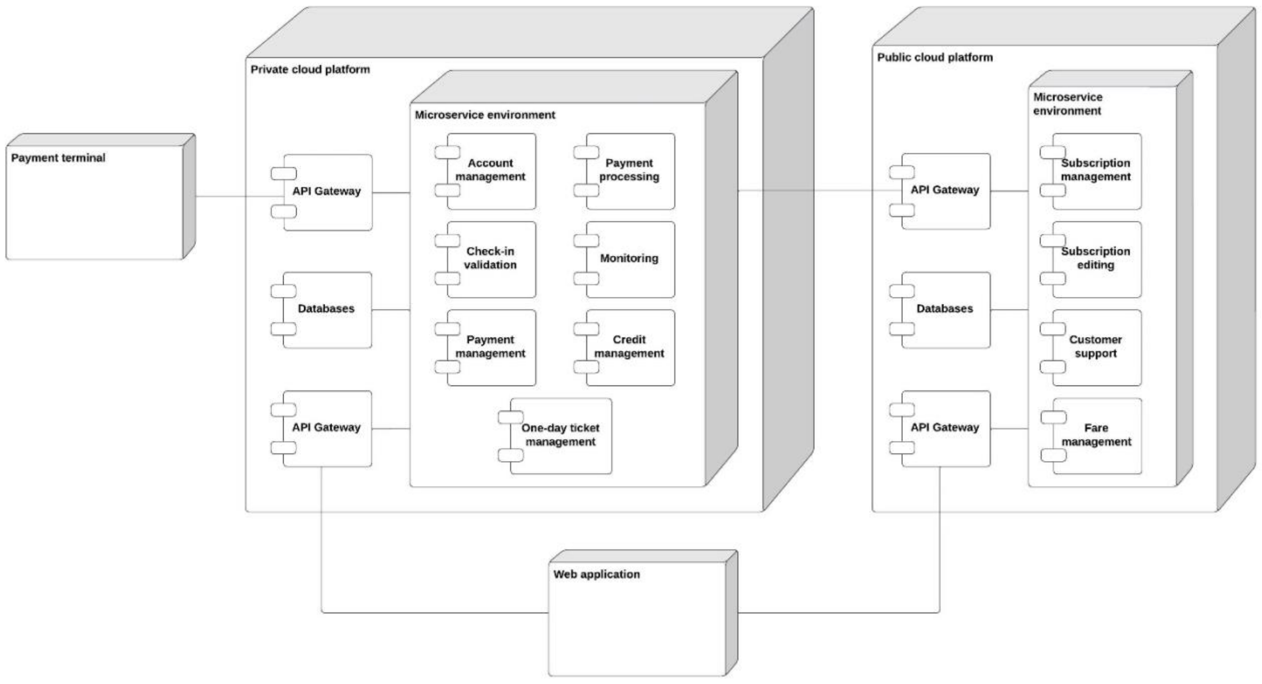

Figure 9

DESCRIPTION

Figure 9 shows the deployment of the TrIP system. The TrIP system is deployed using a hybrid cloud architecture as defined in 'Decision 9: Cloud approach'. The decision covers why hybrid cloud is a well fitting solution for the TrIP system and goes into depth about the functional components and whether they should be deployed in the Private cloud platform or Public cloud platform.

GLOSSARY OF ELEMENTS

Id	Name	Description
1	Payment terminal	The payment terminal is used to check in/out. This information is used by the microservices in Microservice environment on the Private cloud platform.
2	Web application	The Web application is used by passengers and tycoons for all sorts of tasks (specific definition in NOT YET DEFINED). These tasks require a connection to both the Private cloud platform and Public cloud platform.
3	Private cloud platform	The private cloud platform hosts a Microservices environment with microservices or functional components and databases that require tight security.

4	Public cloud platform	The public cloud platform hosts a Microservices environment with microservices or functional components and databases where tight security is not the main concern.
5	API Gateway	Gateways used to communicate between nodes.
6	Databases	All Databases used by the microservices (specific definition on databases is given in the information view)
7	Microservice environment	Place where all microservices of the TrIP system are hosted.

ANALYSIS ON PERSPECTIVES

Describe how the view satisfies the different perspectives.