Software Design & Architecture
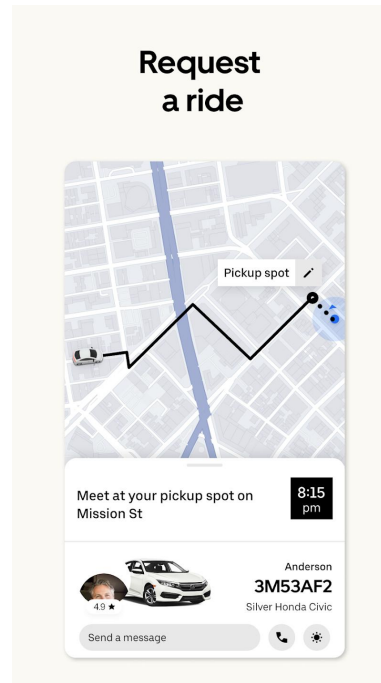
Non Functional Requirements

Agenda

- Functional vs. non-functional requirements
- Types of non-functional requirements

Non-Functional Requirements

- Functional requirements (features)
  - what the system is supposed to do

- Non-functional requirements (constraints)
  - what the system is supposed to be

The app shall allow users to request a ride

The app shall display the driver's location

Request a ride

Pickup spot

Meet at your pickup spot on Mission St   8:15 pm

Anderson
3M53AF2
Silver Honda Civic
4.9 ★

Send a message

Usability: A first-time user shall be able to request a ride in ≤ 3 steps

Efficiency: Driver location shall refresh at least every 2 seconds

FR vs NFR

- Products are sold based on their functional requirements
  - ride sharing, messaging, video streaming; cell phone, car, tent

- However, non-functional requirements play a critical role in perception
  - "This app keeps crashing" (reliability)
  - "It's too slow" (efficiency)
  - "It doesn't work with …" (compatibility)

- Non-functional requirements are differentiators for similar products
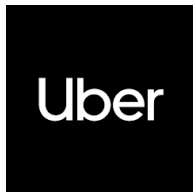
vs

vs

How to write requirements?

- Ask stakeholders / consider the questions they may ask

- Customers: features, user experience

- Management: are we on schedule?

- Developers: who is responsible for implementing what?

- Sales: can we claim it can do this task?

- QA: what teams do we talk to about defects?

- DevOps: where should this component be deployed?

- Support: which QA team signed off on this?

- Maintenance: how can we add this feature?

Types of NFRs

user experience

Usability
Accessibility

performance

Efficiency
Scalability

continuity of service

Availability
Reliability
Robustness
Fault-tolerance
Survivability

harm prevention

Security
Privacy
Safety

ecosystem

Portability
Heterogeneity

*development*

Evovability
Readability
Complexity

NFRs related to User Experience

- **Usability**: How intuitive the user interface of the system is
  - e.g., easily navigate through different features, including X Y Z

- **Accessibility**: The degree to which a product, device, service, or environment is available to as many people as possible
  - e.g., is compatible with screen reader and uses alt text for all images

NFRs related to User Experience - Example

- Transit ticketing app

- The app shall allow riders to plan trips and buy tickets. Riders may be in a hurry, in bright sunlight, with spotty connectivity, and some riders use assistive technologies.

- Usability: A first-time user can buy a single-ride ticket in at most 4 screens, with no account creation required.

- Accessibility: The app supports screen readers, namely all controls have labels, and there is no "color-only" meaning.
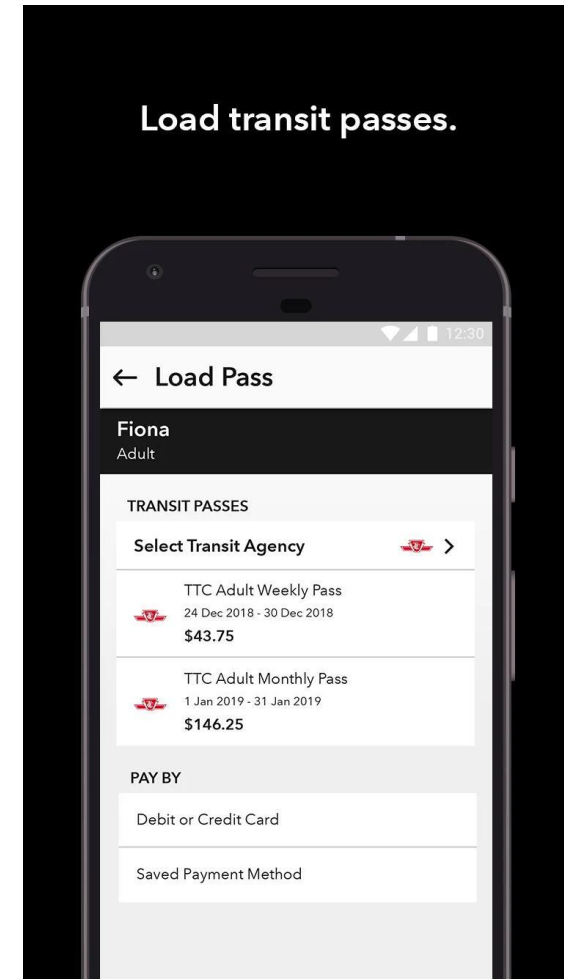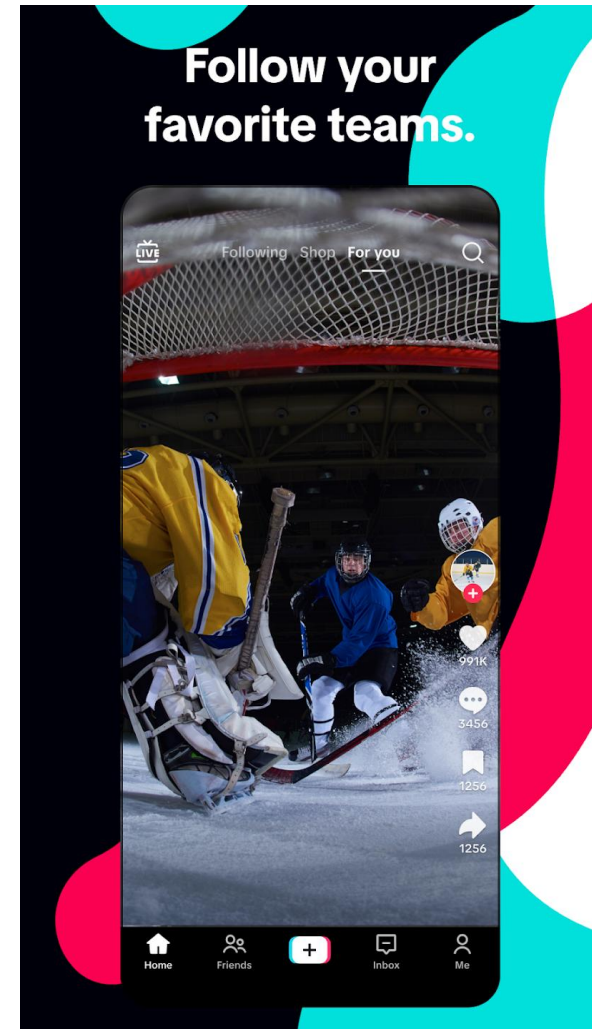
NFRs related to Performance

- **Efficiency**: Ability to meet performance requirements
  - e.g., completes operation Y in X seconds

- **Scalability**: Capability of a system to be adapted to meet new size/scope requirements
  - e.g., can automatically scale resources to support at most X users

NFRs related to Performance - Example

- Short video app

- The app shall allow users to create and upload short videos.
  The home feed shall recommend hot/interesting videos.
  A major event is happening (lots of live users).

- Efficiency (latency): Home feed shall load within 1s, after
  which "time-to-first-content" shall be within 500ms on LTE.

- Efficiency (battery): Background prefetch is capped to
  X MB/day and respects battery saver.

- Scalability: System sustains 20x normal traffic without
 manual intervention, with p95 latency degradation bounded
  to +200ms.

NFRs related to Continuity of Service

normal use

- **Availability**: The probability the system is available at a particular instant in time
  - e.g., up time of XX%

- **Reliability**: The probability that a system will perform within its design limits without failure over time
  - e.g., XX% of success rate

- **Robustness**: Ability to respond adequately to unanticipated runtime conditions
  - e.g., does not crash given the invalid user input of XX

- **Fault-tolenrance**: Ability to respond gracefully to failures at runtime (from environment, components, connectors, component-connector mismatches, etc.)
  - e.g., can continue operate with X number of nodes fail

when things go wrong

- **Survivability**: Ability to resist, recover, and adapt to threats (attacks, failures, accidents, etc.)
  - e.g., redundancy in infrastructure to ensure continuous operation during cyber-attacks or natural disasters

NFRs related to Continuity of Service - Example

- Ride sharing app

- Availability: The "request ride" core feature shall has 99.9% monthly availability.

- ...at peak times (e.g., on New Year's Eve)

- Robustness: The app shall handle intermittent connectivity without crashing; there should be clear notifications.

- Fault-tolerance: If one map provider fails, fall back to another or to cached maps.

- Survivability: During partial outage, preserve essential service and allow ongoing rides to complete.
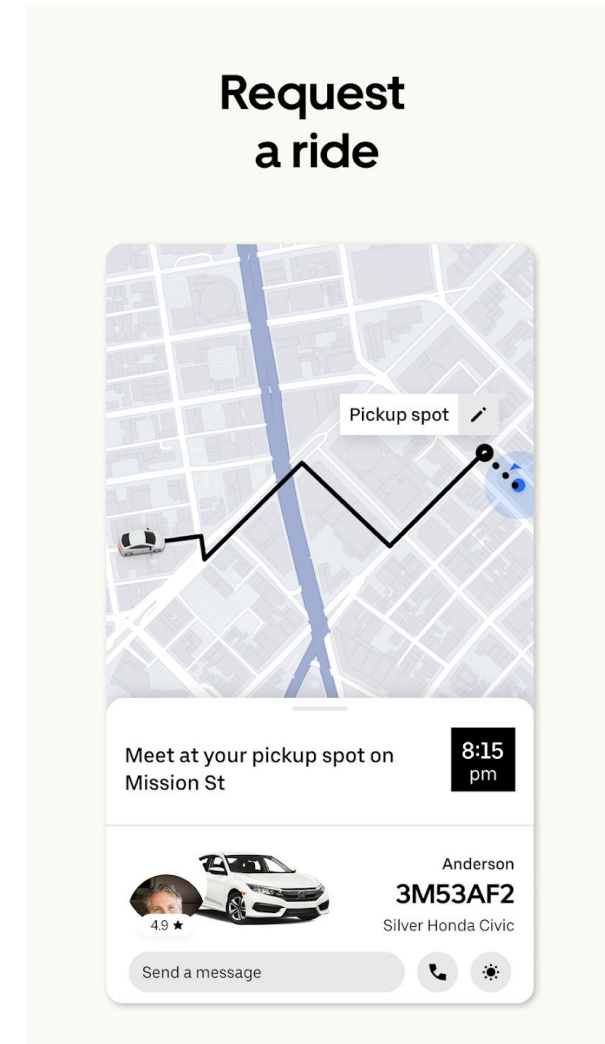
NFRs related to Harm Prevention

- **Security**: How well the system protects users from external attacks
  - e.g., multi-factor authentication to protect user accounts from unauthorized access

- **Privacy**: How a system protects the private information of the user
  - e.g., end-to-end encryption for private messages

- **Safety**: Ability to avoid failures that will cause loss of life, injury, or loss to property
  - e.g., collision detection in autonomous driving system

NFRs related to Harm Prevention

- Online banking app

- Threats can include credential theft, device loss, man-in-the-middle attacks, fraud, ...

- Security: The app shall require 2nd-factor authentication (message/phone) for high-risk actions.

- Privacy: The app shall acquire user consent before collecting transactions data for analytics.

- Safety: For transfers to a new payee, the app shall enforce a cooling-off period before the transfer can be completed.
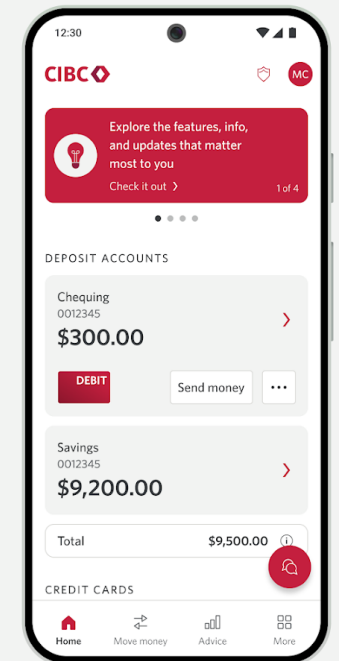
NFRs related to Ecosystem

- **Portability**: Ability to execute on multiple platforms while retaining their functional and non-functional properties
  - e.g., runs on Windows, MacOS, and Linux

- **Heterogeneity**: Ability to be composed of, or execute within, disparate parts
  - e.g., consists of X modules implemented in programming languages A B C correspondingly
  - e.g., (fitness tracking app) works across diverse sensors and handles missing sensors gracefully

NFRs related to Development

- **Evolvability**: Ability to react on change, satisfy new requirements, and add support for new environments
  - e.g., features can be easily added/modified under the modular architecture

- **Readability**: How well the system is comprehensible to a new developer
  - e.g., every function should have no more than X lines and have comments

- **Complexity**: The size of a system, the volume of constituent elements, their internal structure, and their interdependencies
  - e.g., consists of X modules, each with around Y lines of code

Exercise: NFRs

- Waterloo Learn course management system

- Discuss and come up with NFRs
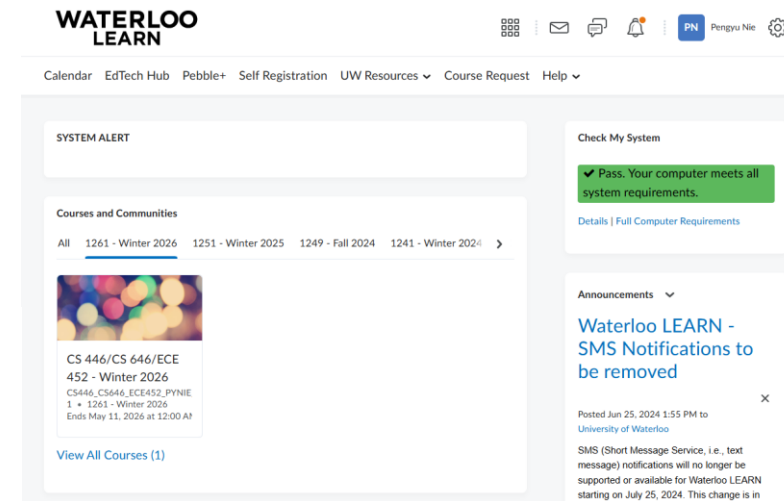
user experience

Usability
Accessibility

performance

Efficiency
Scalability

continuity of service

Availability
Reliability
Robustness
Fault-tolerance
Survivability

harm prevention

Security
Privacy
Safety

ecosystem

Portability
Heterogeneity

development

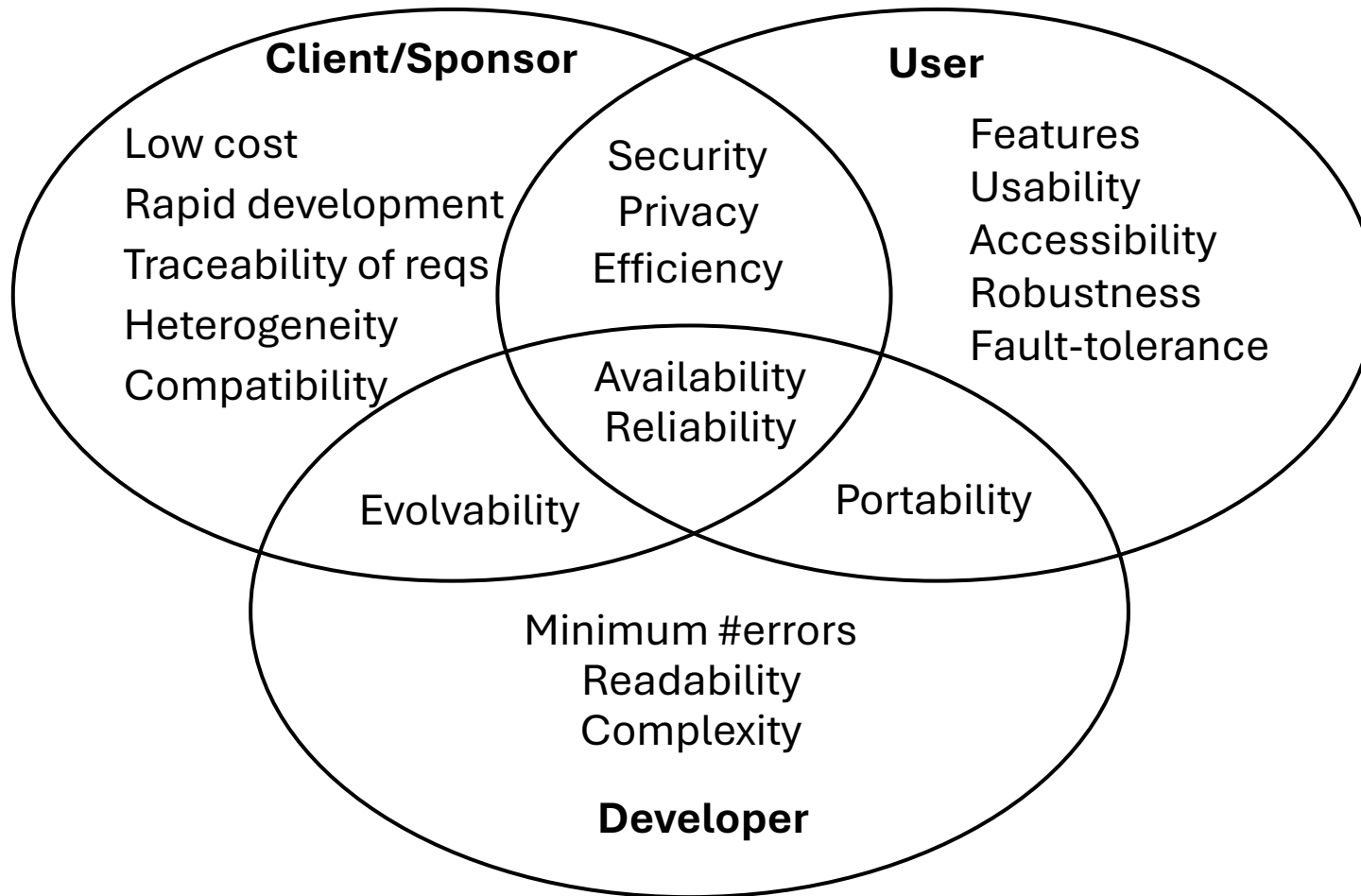| Evovability |
|---|
| Readability |
| Complexity |

Evaluating NFRs

- Think about NFRs concretely
  - how can they be measured?
  - use specific numbers/cases/items

	Good	Bad
Efficiency	The system shall issue new tickets in under 10 seconds	The system shall issue tickets quickly
Usability	The system shall enable a user with no training to buy tickets in four clicks or less	The system shall be user-friendly

Stakeholder Conflicts on NFRs

- Each stakeholder will have their own opinion about what (non-functional) requirements matter most

Client/Sponsor

Low cost
Rapid development
Traceability of reqs
Heterogeneity
Compatibility

Security
Privacy
Efficiency

User

Features
Usability
Accessibility
Robustness
Fault-tolerance

Availability
Reliability

Evolvability

Portability

Minimum #errors
Readability
Complexity

Developer

Typical Tradeoffs

- Security vs. Usability

- Functionality vs. Usability
  - lots of features vs. simple flows

- Evolvability vs. Cost
  - reusable components vs. one-off delivery

- Efficiency vs. Portability
  - native optimizations vs. runs everywhere

Recap

- Functional vs. non-functional requirements

- Types of non-functional requirements

- Conflicts & tradeoffs

user experience
| Usability |
| Accessibility |

performance
| Efficiency |
| Scalability |

continuity of service
| Availability |
| Reliability |
| Robustness |
| Fault-tolerance |
| Survivability |

harm prevention
| Security |
| Privacy |
| Safety |

ecosystem
| Portability |
| Heterogeneity |

development
| Evovability |
| Readability |
| Complexity |

- Reminder: P0 team formation due this Friday