

Deliverable 1

Group: **The Architects**

Members: **Jacob Grum, Rasmus Ole Routh Herskind & Thor Liam Møller Clausen**



System description


Overall functionality




Imagine the following; Kevin wants to bet on Holger winning his tennis match. He checks his usual site, which gives odds 1.8. But his mate next to him scuffs at him “you should check out my site, it gives odds 1.9”. Kevin quickly realizes that he is not loyal to his site, he is mostly interested in placing on any site with the best odds available - *Introducing **BestBets***:

Shows the best odds available on any site, for any match.


All sport



☐ Toggle community-bets

 [Barcelona - Arsenal: 1x2](#)

1: Barcelona	x: Draw	2: Unibet
 4,75	 13,63	 10,00

[See bets for all sites](#)


 [Holger Rune - David Goffin](#)

1: Holger Rune	2: David Goffin
 4,75	 13,63

[See bets for all sites](#)

Plenty of stock-like features! Trigger order, when a certain game reaches certain odds.

Bets-børsen

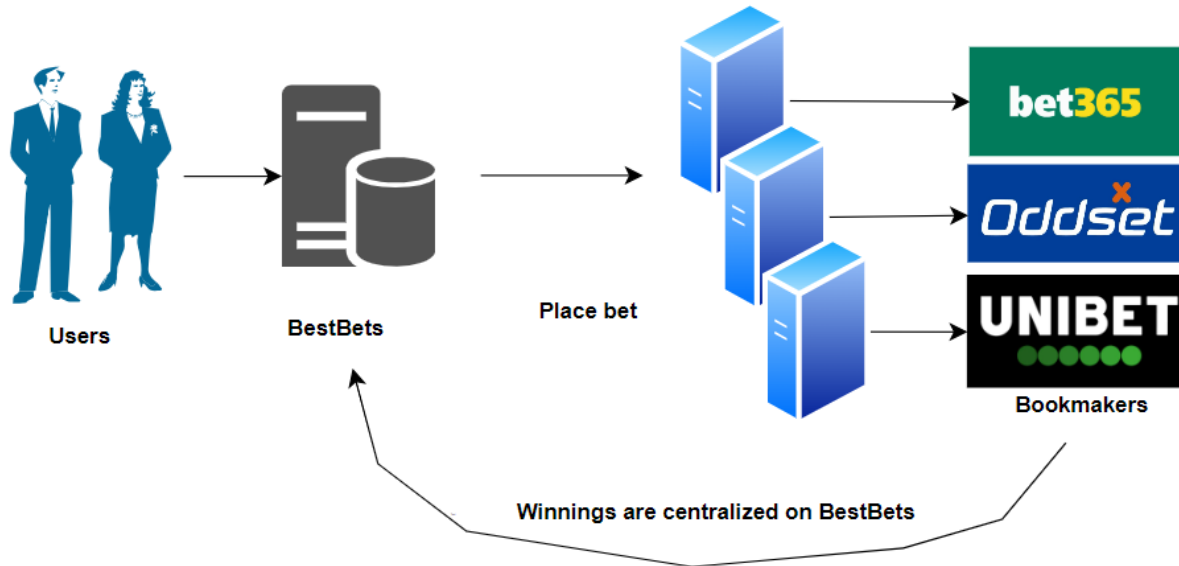
 [Barcelona - Arsenal: 1x2](#)

Automatically place bets on best site, if odds exceeds X

1: Barcelona	x: Draw	2: Unibet
<input type="text" value="Number-input"/>	<input type="text" value="Number-input"/>	<input type="text" value="Number-input"/>

Bet size


Similar market conditions as stocks. You don't want to search for the best offer from a bank, you trade on a shared market (Børsen). That means you only need to have money in one central place, to be able to purchase from all bookmakers! In short. If you have 200 kr on BestBets, you can place 200 kr on ANY bookmaker, without having a saving of 200 kr on each bookmaker.



Similar to shorting a stock. Think the best odds are bad? Open your own position, and act like the bookmaker!

Create community-bet

Community-bets must be greater than odds given by any vendor

 [Holger Rune - David Goffin](#)


1: Holger Rune


2: David Goffin


Accepted bet amount
[Number-input]


Quickly register for any bookmaker, through a saved formula, to get consistent information across multiple sites, and to always be ready to place on the best odds possible.


Register


☐


☐


☒


☒

☐

☒

☒

☐

☐


⋮


Start registering


→

Desired account information

☒ I want same information on all sites*

 Full name

 User name

 Password

⋮

☐ I want to chose individually on each site

Create accounts

* Form is saved to primary browser.
So it can autofill the native forms of the website





Get an instant overview of current bets, accounts and bonuses!

Overall amount: 2500 kr.

Withdrawable: 1750 kr.

Bonuses: 750 kr

Specific account overview

Connected accounts	Active site specific bonus amount	Current bets placed
	81 kr.	You have 0 active bets
	41 kr.	You have 4 active bets
	123 kr.	You have 0 active bets
	500 kr	You have 2 active bets

[Connect more accounts](#)

Connect already existing accounts

Register new sites, and connect directly

Various settings to ensure only your favorite sports are shown, to ease navigation and usage.

Show currency in: ▼

Block these sites ▼

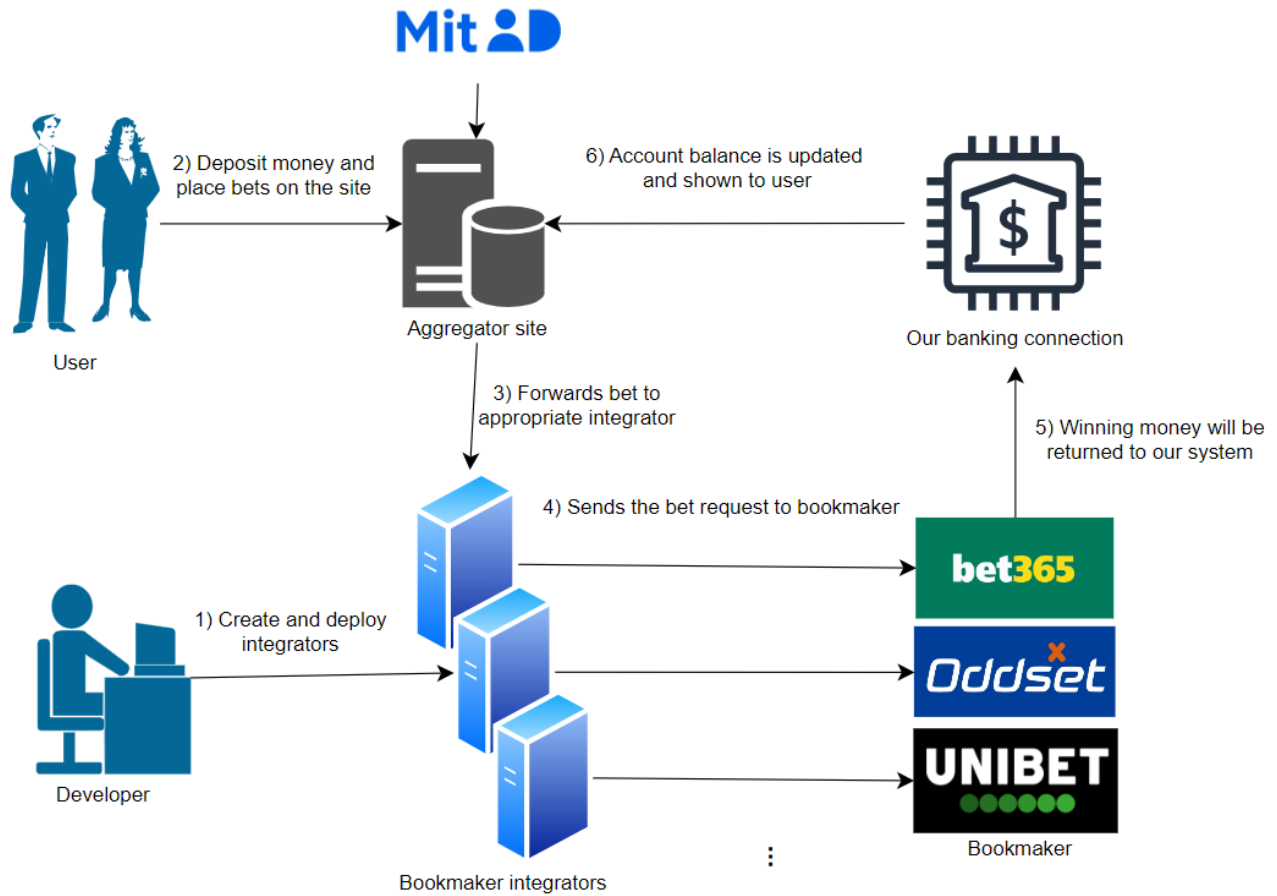
Show community-bets ▼

Show odds as:
2.15
or
1:2.15 ▼

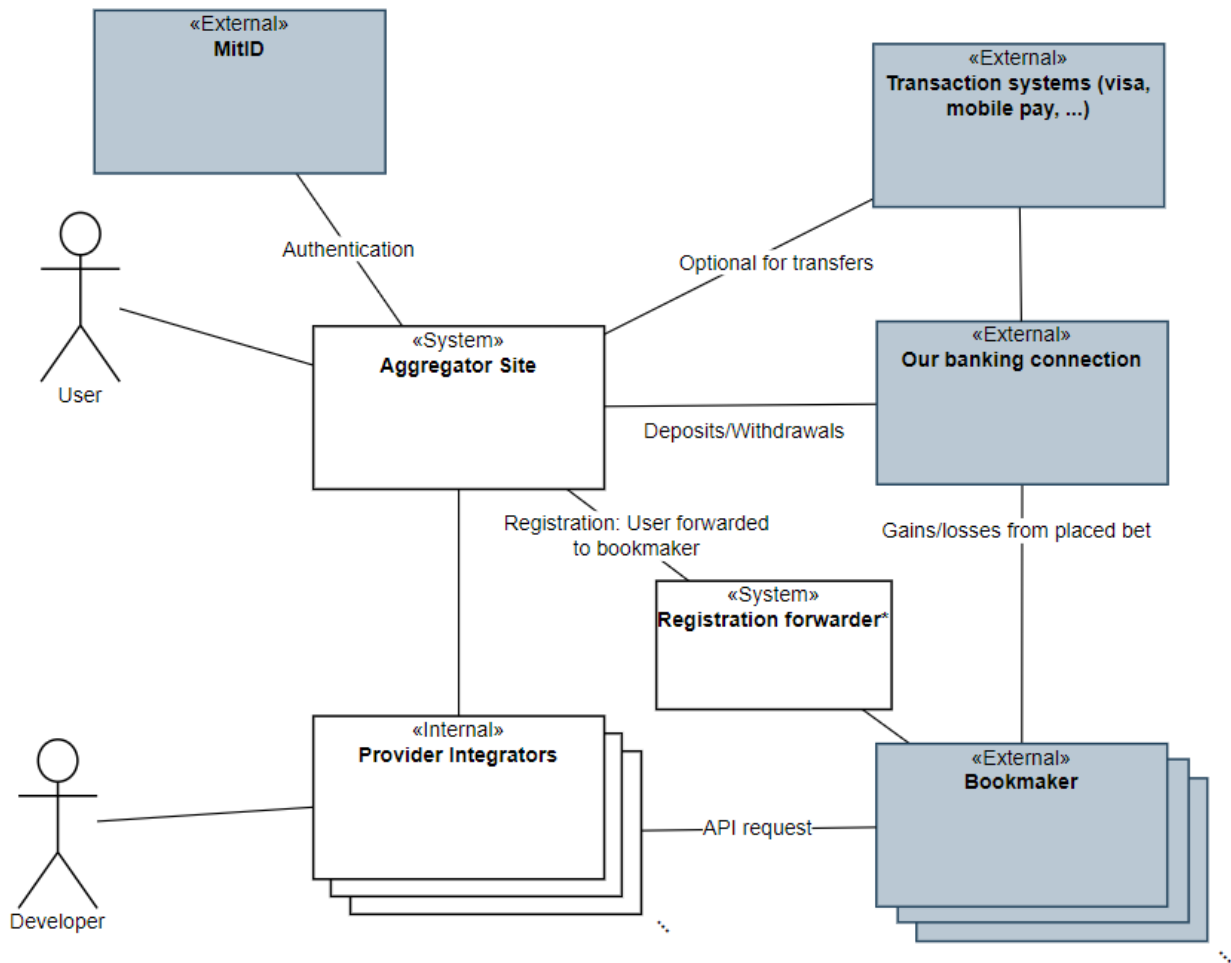
...

Context diagrams

Informal diagram



Formal diagram



*Since we cannot register users through API we need to redirect users to the Betting site for this part

Architectural analysis

Identification of Architectural Drivers

The numbers in parenthesis is the number of points given in our stakeholder role play.

- **Modifiability Requirement:**
 - **Handle new bookmakers (11)**

The architecture shall be modifiable so that when a new bookmaker becomes available or an existing one ceases operation, support for that bookmaker can be added or removed with minimal impact on the overall system.
- **Availability Requirement:**
 - **Available during high demand (12)**

The system must remain available even during periods of high user demand (e.g., during high-traffic sporting events) to ensure continuous access to betting data.
- **Performance Requirements:**
 - **Low Latency (14)**

The system shall fetch and display betting odds from multiple providers in real-time or near-real-time. Delays in page loads or data updates must be minimized to prevent users from missing favorable odds.
 - **Efficient Data Processing (14)**

The system must process large volumes of betting data rapidly (especially during peak activity such as major finals) so that performance remains robust under load.
- **Safety Requirement:**
 - **Safely handle odds changes (10)**

The system shall ensure safety during the bet placement and acceptance process. If betting odds change between the time a bet is placed and accepted, the system must reject the altered odds and prompt the customer to re-confirm their bet under the new conditions.
- **Usability Requirements:**
 - **Intuitive Navigation (12):**

Users shall be able to quickly and easily find the best odds for a match.
 - **Responsive Design (11):**

The interface must adapt seamlessly to various devices (mobile, tablet, desktop) to accommodate users checking odds on the go.
 - **Support for Specific Scenarios (12):**

Filtering. Only tennis, only matches in Austrilian Open, that ends within 12 hours.
 - **Consistent design (12).** A bet placed on bookmaker A must have same mechanics and appearance as on bookmaker B.

- **Security Requirement:**

- **The system must ensure robust security (9):**

- User data and communications must be safeguarded using secure handling mechanisms.

- Given that customers place real money on bets, the system must ensure that all financial and personal data are protected from unauthorized access.

Most important ones:

1. Performance (14)

- Low latency: Provide fresh odds.
- Efficient Data Processing: During many bets placed during big event

2. Availability(13)

- System must work and respond during high traffic for big events

3. Usability (12)

- Intuitive design. Our aggregator must provide an interface that does not get affected by what bookmaker the bets are offered from. Betting on A should look the same as betting on B, where A and B are two different bookmakers.

4. Modifiability (11)

- In case many new bookmakers open or others close, it must be handled without too much trouble.

5. Safety (10)

- With money being handled, the user must not be prone to make mistakes.

Stakeholder roles

Often directly in touch with the system

- Betting sites, also referred to as “Bookmakers”
- Developers/Architects
- Users

Various gambling related governmental instances

- Spillemyndigheden: <https://www.spillemyndigheden.dk/gaming-og-gambling>
- Ludomani støtteforeninger: <https://www.stopspillet.dk/>

Various partners and affiliates:

- Sport-event industry (Arenas, leagues, etc).
- Sport clubs
- Sports news (magazines, blogs, etc.)
- Sports apparel and merchandise
- Sport streaming & channels

Quality Attribute Scenarios

Performance

Refined Scenario 1: Real-Time Betting Odds Update

Scenario: The system updates betting odds in real-time during a high-profile sports event.

Relevant quality: Performance

Stimulus source: External betting data providers.

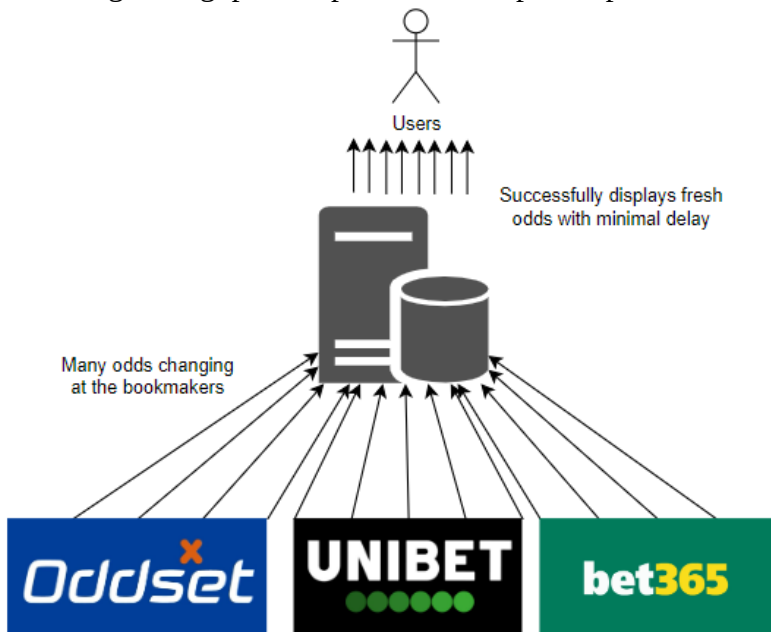
Stimulus: A sudden surge in updated odds from multiple providers as a major match reaches a critical point.

Environment: System is running high traffic during live, high-stakes sports events.

Artifact (if known): The module(s) that fetches live data from betting sites and updates our site.

Response: Fetch and process incoming data rapidly and update the user interface with minimal delay.

Response measure: Updated odds are displayed within 500 milliseconds of detected change, sustaining throughput of up to 100.000 updates per minute without performance degradation.



Refined Scenario 2: Massive user surge during the Champions League final

Scenario: The system responds to a lot of users during Champions League final with little delay.

Relevant quality: Performance/Scalability

Stimulus source: A lot of end users (100.000) logging in at the same time.

Stimulus: Number of users logged in increases from 1000 to 100.000 in the span of 10 minutes.

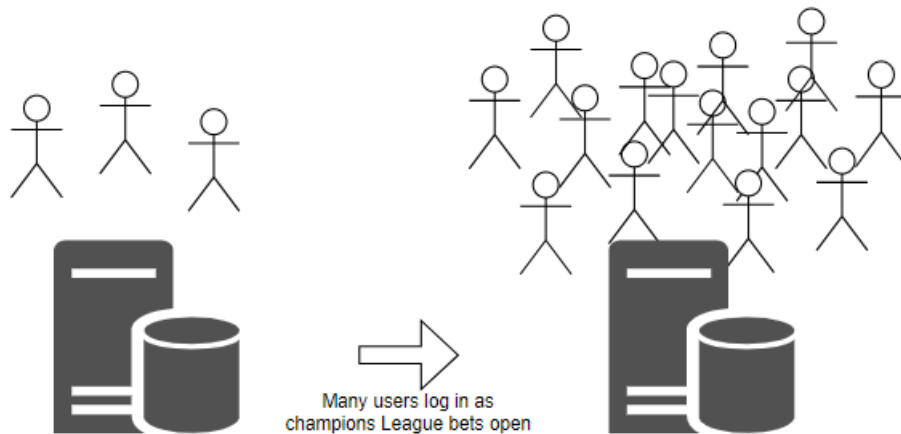
Environment: System running normally.

Artifact (if known):

Response: The system continues to serve updated odds without errors or significant delays

Response measure: Pages load within 2 seconds for 95% of requests. Odds update within 3

seconds of provider feed changes.



Refined Scenario 3: Massive surge of placed bets during prep for a mayor sports match

Scenario: The systems handles a lot of bets placed within a short time span with little delay.

Relevant quality: Performance

Stimulus source: A lot of end users (100.000) placing bets.

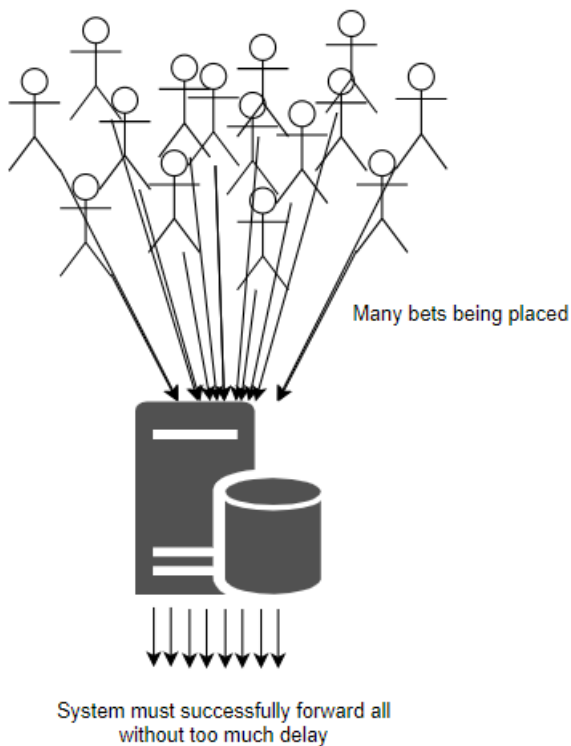
Stimulus: 100.000 bets being placed per 5 minutes.

Environment: The system running at high capacity with many users logged in.

Artifact (if known): The module(s) handling the forwarding of bets to the betting sites, and module(s) receiving confirmation from the sites.

Response: The system successfully forwards the bet placements without creating bottlenecks.

Response measure: With 100.000 bets per 5 minutes the average latency should be below 1 second from user confirms in our system to the bet being successfully forwarded to the respective betting site.



In short:

- 1: Handle external updates of odds.**
- 2: Many users logs into our internal site.**
- 3: Many bets are placed.**

Availability

Refined Scenario 1: Continuous Availability Under Traffic Spike

Scenario: The platform remains fully available during an unexpected surge in user traffic.

Relevant quality: Availability

Stimulus source: End users accessing the site during a major sporting event.

Stimulus: 100.000 users simultaneously accessing and interacting with the platform.

Environment: System running on high capacity from the many logged in users.

Artifact (if known): The modules handling user activity (front end, server, load balancers, auto-scaling infrastructure).

Response: The system dynamically scales resources and distributes the load to maintain service availability.

Response measure: Achieves 99.9% uptime during peak load with 95% of requests served in under 500 milliseconds.

Refined Scenario 2: Site remains available during betting site outages

Scenario: The system handles the failure of one or more betting sites without affecting overall availability.

Relevant quality: Availability

Stimulus source: Betting sites failing to respond to requests (ping).

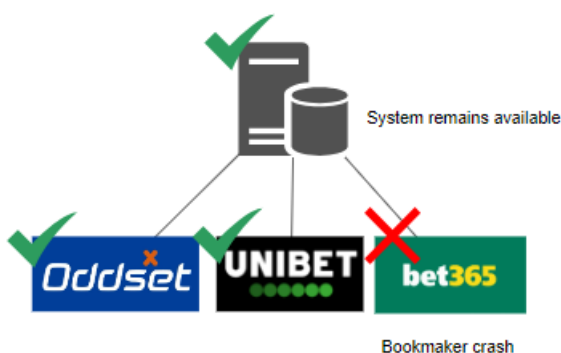
Stimulus: Ping requests without a response within 2 seconds from one or more betting sites.

Environment: System running normally with logged in end users.

Artifact (if known): Modules communicating with betting sites (fetching data, checking for activity).

Response: The system continues to display odds from other providers and clearly labels outed providers as “unavailable”.

Response measure: The system has a 99.99% uptime with up to 100% of betting sites unavailable.



Refined Scenario 3: The integration of a new betting site does not affect availability

Scenario: A new betting site has been integrated in the system and gets deployed without affecting active users.

Relevant quality: Availability

Stimulus source: The developers deploying a new betting site opportunity in the system.

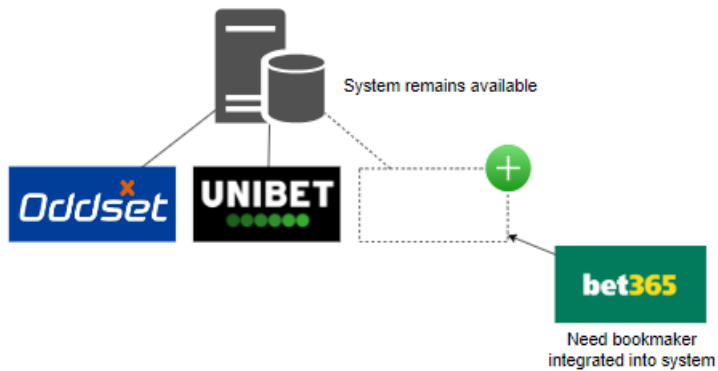
Stimulus: Initiation of the deployment of the new betting site integration.

Environment: System running normally with logged in end users.

Artifact (if known): The modules interacting with the betting site integrations/the aggregator.

Response: The system switches seamlessly to the updated version without service disruption.

Response measure: Zero downtime is observed; the switchover is completed seamlessly with monitoring confirming uninterrupted service.



Usability

Refined Scenario 1: Consistent Navigation Across Devices

Scenario: The system provides a uniform and consistent user experience across various devices.

Relevant quality: Usability

Stimulus source: End users accessing the system on desktops, tablets, and mobile devices.

Stimulus: A user transitions between devices and expects the interface to behave consistently.

Environment: System running normally.

Artifact (if known): User interface.

Response: The user interface maintains consistent layout, design, and navigation regardless of the device.

Response measure: Less than 15% difference in the speed of user interaction across device.

Refined Scenario 2: Intuitive Filtering and Search

Scenario: Users quickly locate and filter betting events to find their preferred matches.

Relevant quality: Usability

Stimulus source: End users utilizing search and filtering tools.

Stimulus: A user applies filters (e.g., for tennis matches or events ending within 12 hours) to quickly locate relevant betting options.

Environment: System running normally with search/filter options implemented.

Artifact (if known): UI components and logical components managing search and filtering.

Response: The system delivers relevant results promptly, making it easy for users to narrow down options.

Response measure: At least 90% of search queries return results within 300 milliseconds; user feedback on the filtering process is overwhelmingly positive (95% satisfaction).

Refined Scenario 3: Streamlined, Consistent Bet Placement Process

Scenario: The bet placement workflow is intuitive and consistent across different sections of the site.

Relevant quality: Usability

Stimulus source: A user wanting to place a bet on several different betting sites using our system.

Stimulus: The user navigating through the system to place their bets.

Environment: System running normally.

Artifact (if known): The UI allowing for bet placement.

Response: The user successfully placing their bets without much delay in between bets.

Response measure: 90% of users successfully placing bets in less than 2 minutes on average and in case of multiple bets, at most 1 minute between bets on average.

Modifiability

Refined Scenario 1: Adding a New Betting Provider

Scenario: The system integrates a new betting provider with minimal disruption.

Relevant quality: Modifiability

Stimulus source: A business decision to expand service offerings.

Stimulus: Announcement of a new betting provider entering the market.

Environment: Development and production environments.

Artifact (if known): Aggregator codebase (API integration modules)

Response: Development team implements the new provider interface, tests it, and deploys it with minimal disruption, and minimal changes to the existing code base.

Response measure: Completed integration within 5 developer days. No existing functionality broken (automated tests pass). Less than 5% of the existing code base changed.

Refined Scenario 2: Removing an Existing Betting Provider

Scenario: The system gracefully removes support for a provider that is ceasing operations.

Relevant quality: Modifiability

Stimulus source: External betting site shutting down.

Stimulus: An existing betting provider has been decided to be removed from the system.

Environment: The system running with multiple integrated betting sites.

Artifact (if known): The modules interacting with the betting site integrations/the aggregator.

Response: The betting site integration is isolated for safe removal.

Response measure: Integration removal is executed in under 24 hours with zero system downtime.

Refined Scenario 3: Adapting to Changing Provider APIs

Scenario: The system is modified to handle betting provider's changed API with minimal impact on overall functionality.

Relevant quality: Modifiability

Stimulus source: External betting provider API.

Stimulus: The betting provider releases a new version of its API.

Environment: System running with multiple integrated betting providers.

Artifact (if known): API adapter for external services.

Response: The API adapter is updated to accommodate the API changes while keeping the core system unaffected.

Response measure: The required modifications are completed and fully tested within 48 hours with only localized changes in the adapter code.

Safety

Refined Scenario 1: Acceptance of changed odds

Scenario: There is a difference in the odds between the time the user enters its desired bet, and when the bet is being confirmed at the vendor

Relevant quality: Safety/Usability

Stimulus source: External betting provider.

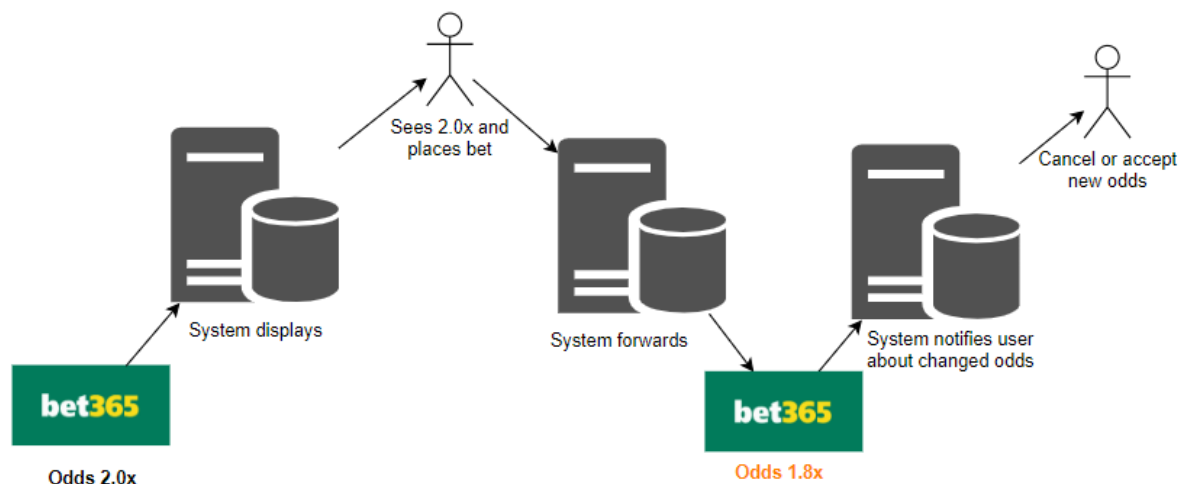
Stimulus: A user enters a bet and, immediately afterward, the betting odds change before confirmation.

Environment: System is running normally

Artifact (if known): Module(s) handling placement of bets.

Response: The initial desired bet is rejected, and user is asked if they want to place a new bet at the changed odds.

Response measure: 100% of bets affected by odds changes are safely rejected and re-confirmed, with user prompts issued within 1 second of detecting the change.



Refined Scenario 2: Impossible bets can't be placed.

Scenario: A user enters a desired bet, on a site that between confirmation and placement, gets removed from the external vendor.

Relevant quality: Safety

Stimulus source: External betting provider.

Stimulus: A user enters a bet and, immediately afterward, the bet is completely unavailable

Environment: System is running normally

Artifact (if known): Module(s) handling placement of bets.

Response: The initial desired bet is rejected. User is warned that no bet has been made, and is presented the highest available bet from another vendor.

Response measure: 100% of bets affected by removal are safely rejected, with a warning issued within 1 second of detection.

Refined Scenario 3: External betting site errors does not cause faulty behavior

Scenario: A betting provider is responding with error messages or not at all.

Relevant quality: Safety

Stimulus source: External betting providers.

Stimulus: Error responses when trying to communicate with betting provider.

Environment: System running normally.

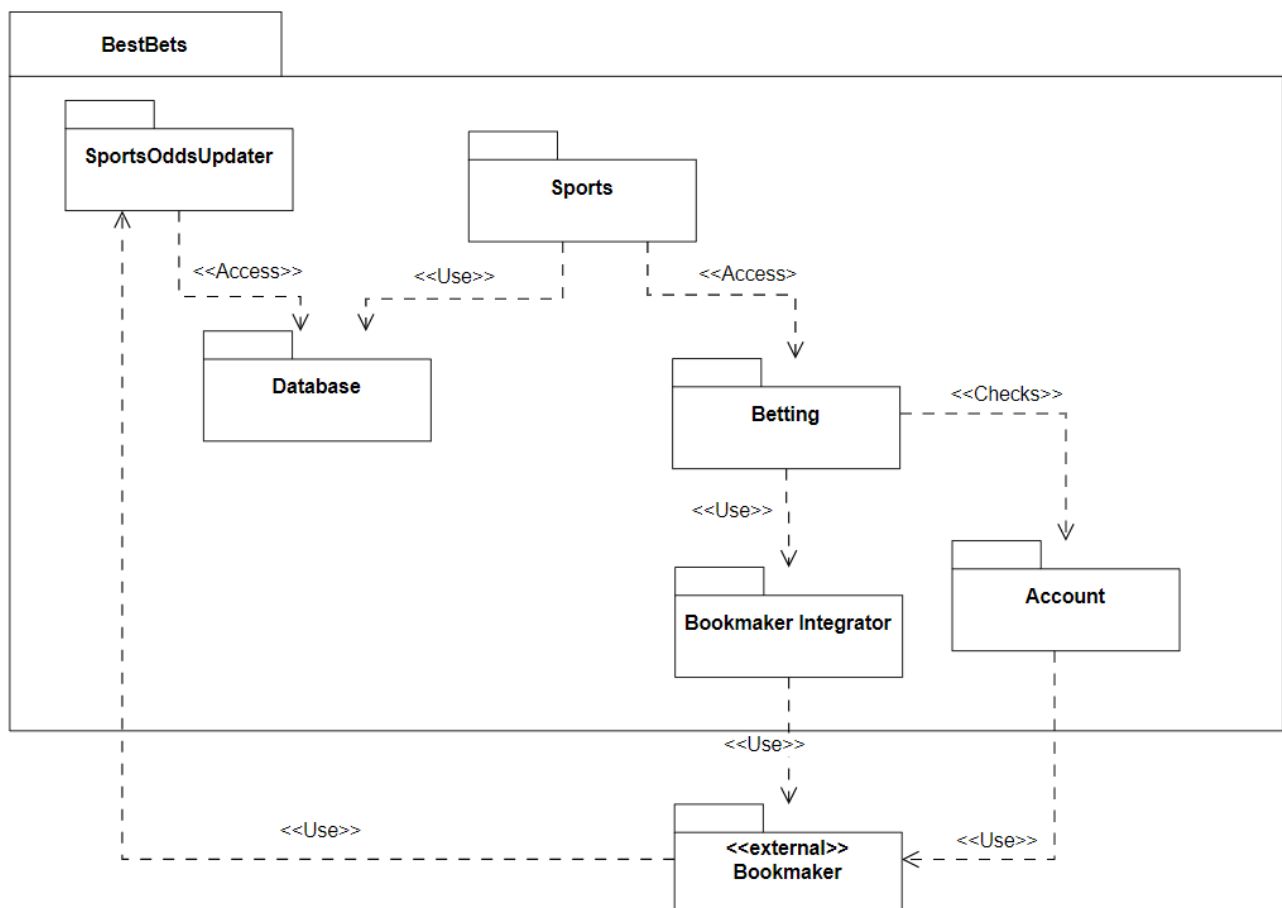
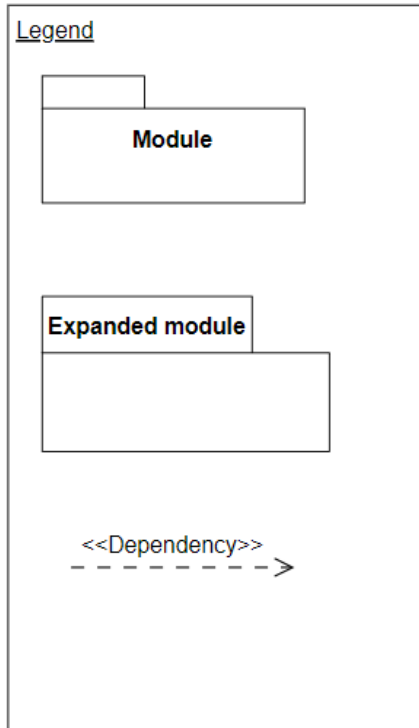
Artifact (if known): Model(s) interacting with external betting providers.

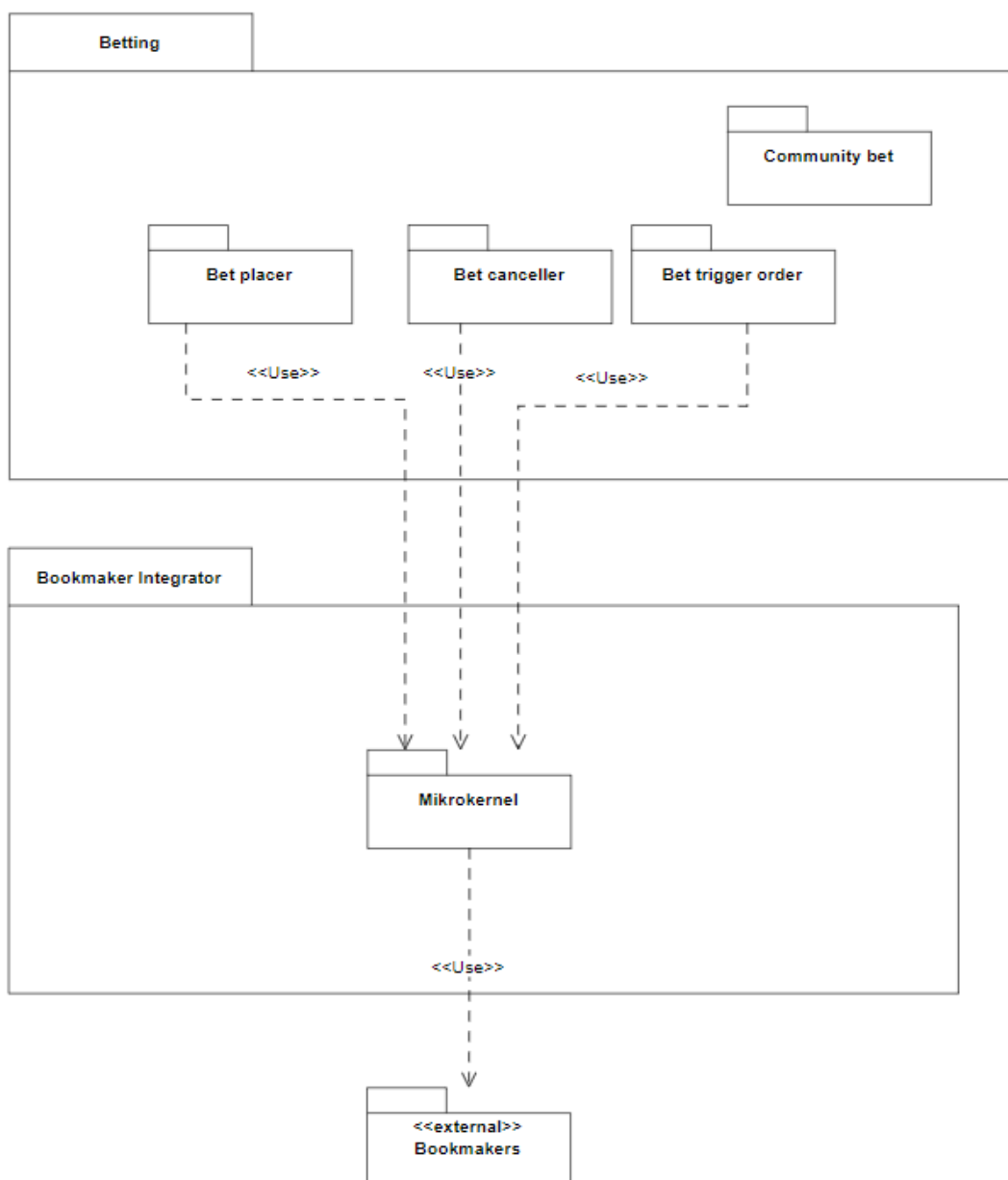
Response: Our system continues to run without errors or faulty behaviour.

Response measure: 0 error detected in our system in the case of errors from external providers.

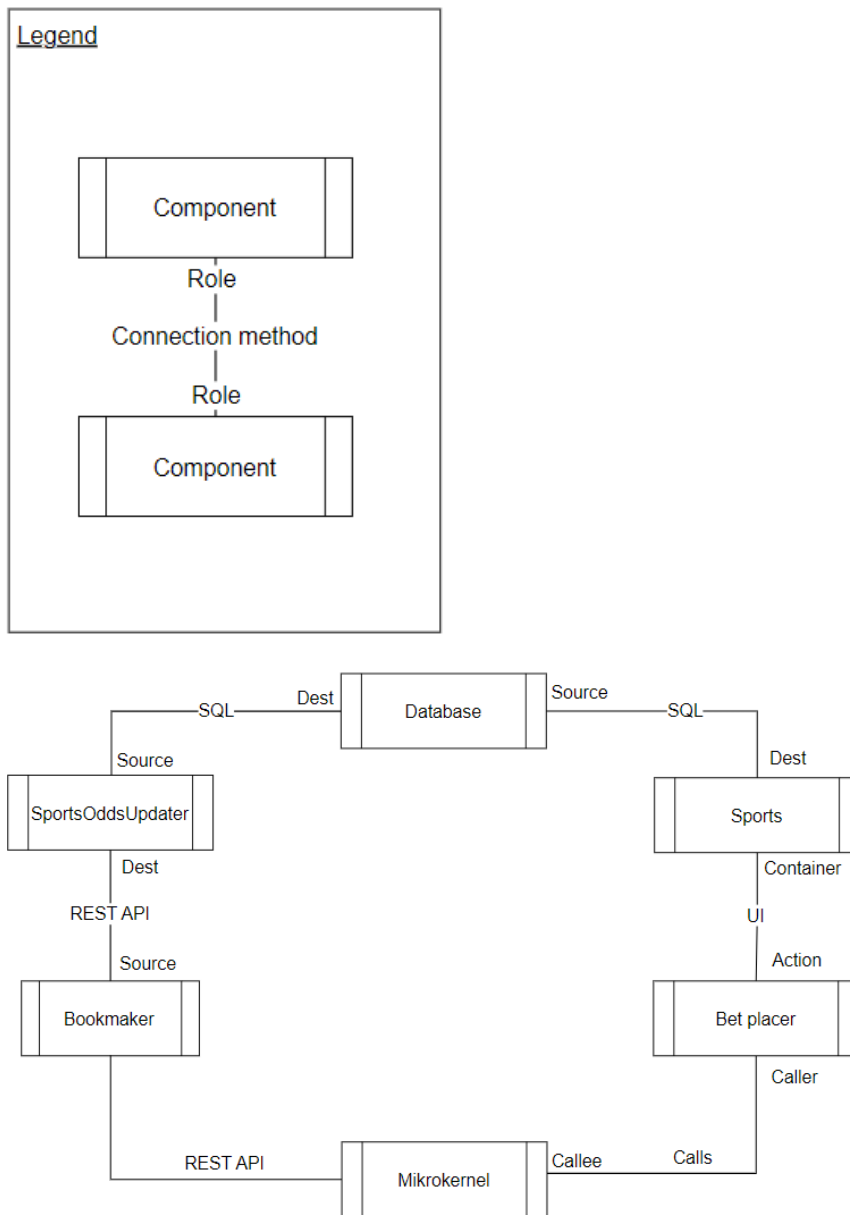
Architectural synthesis

Module view

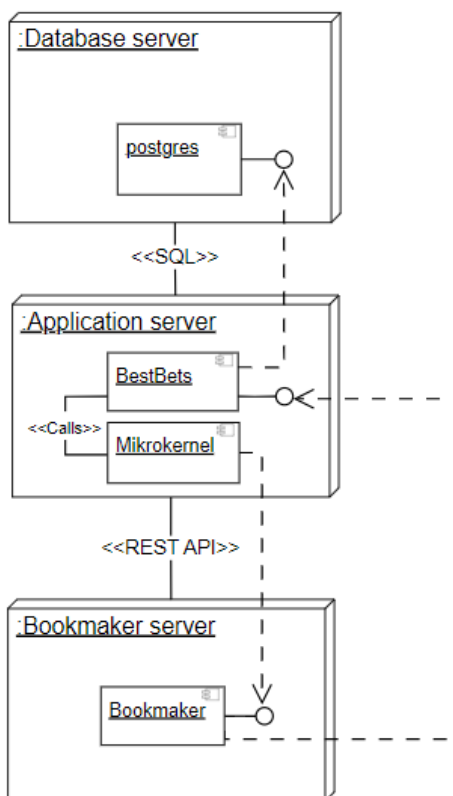
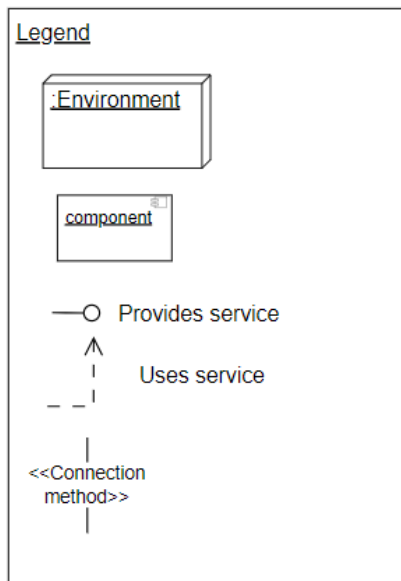




Component & connector view



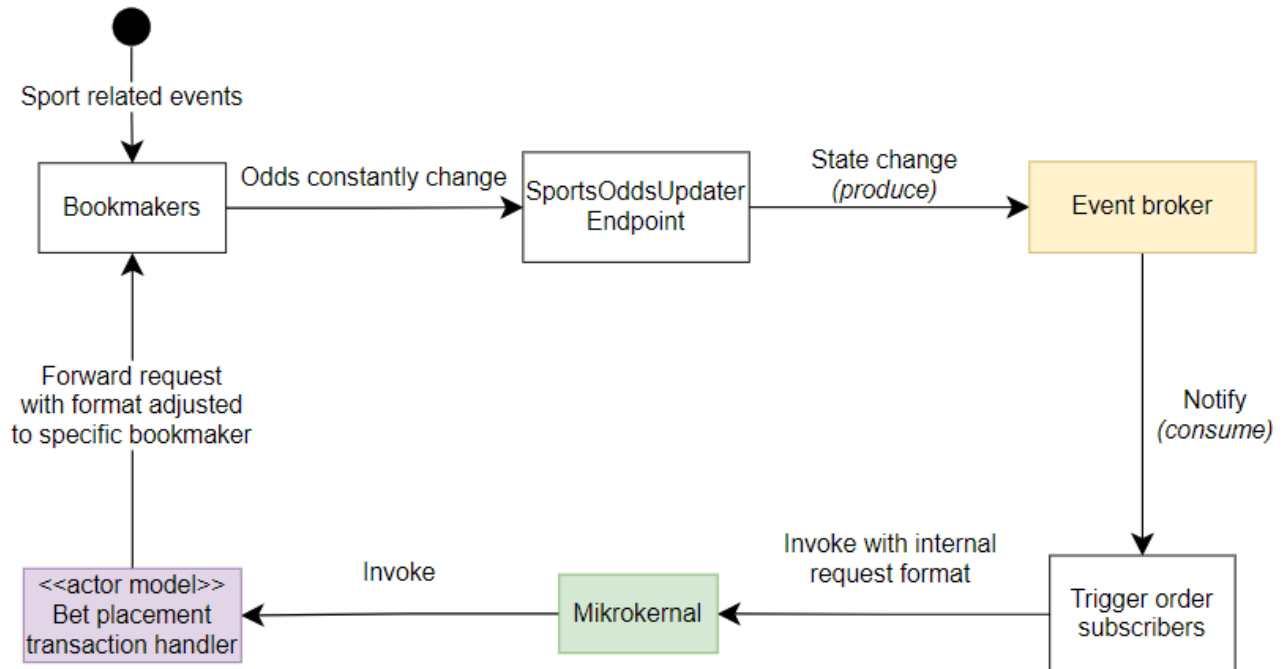
Deployment view



Design decisions regarding Tactics & Patterns

Overall structure

Zoom: 1



Event broker zoomed in view

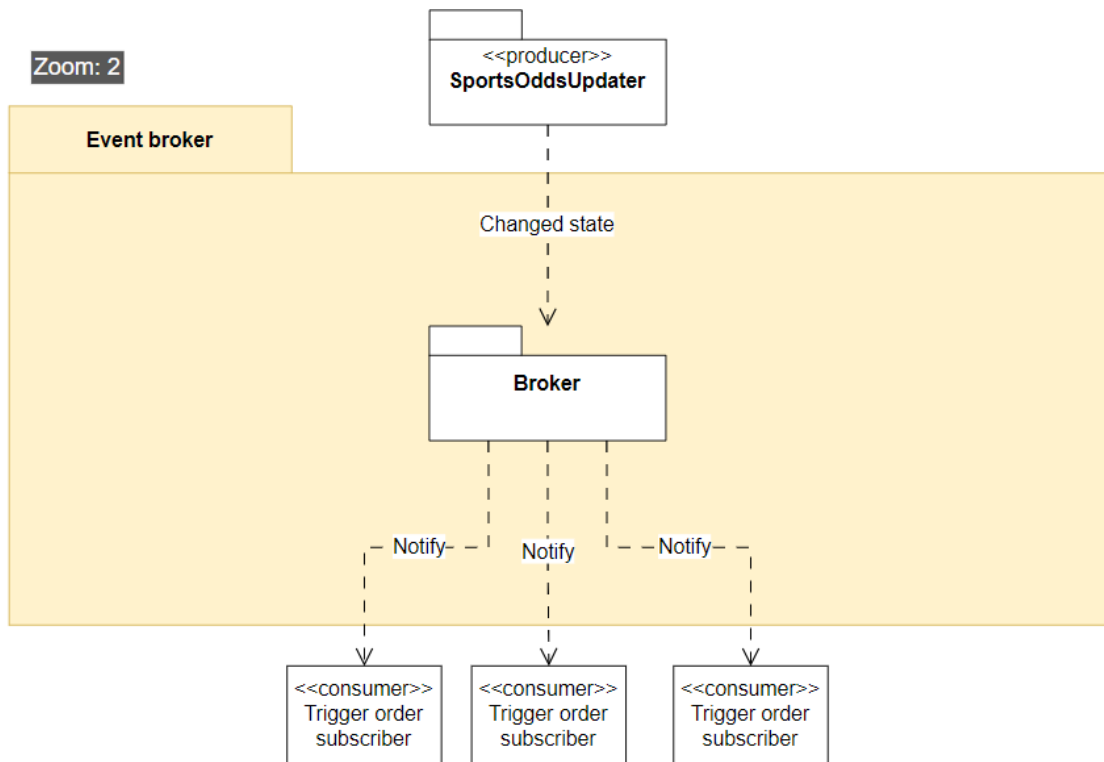


Figure 1: What happens inside the event broker

Microkernel zoomed in view

Zoom: 2

External betting API handler

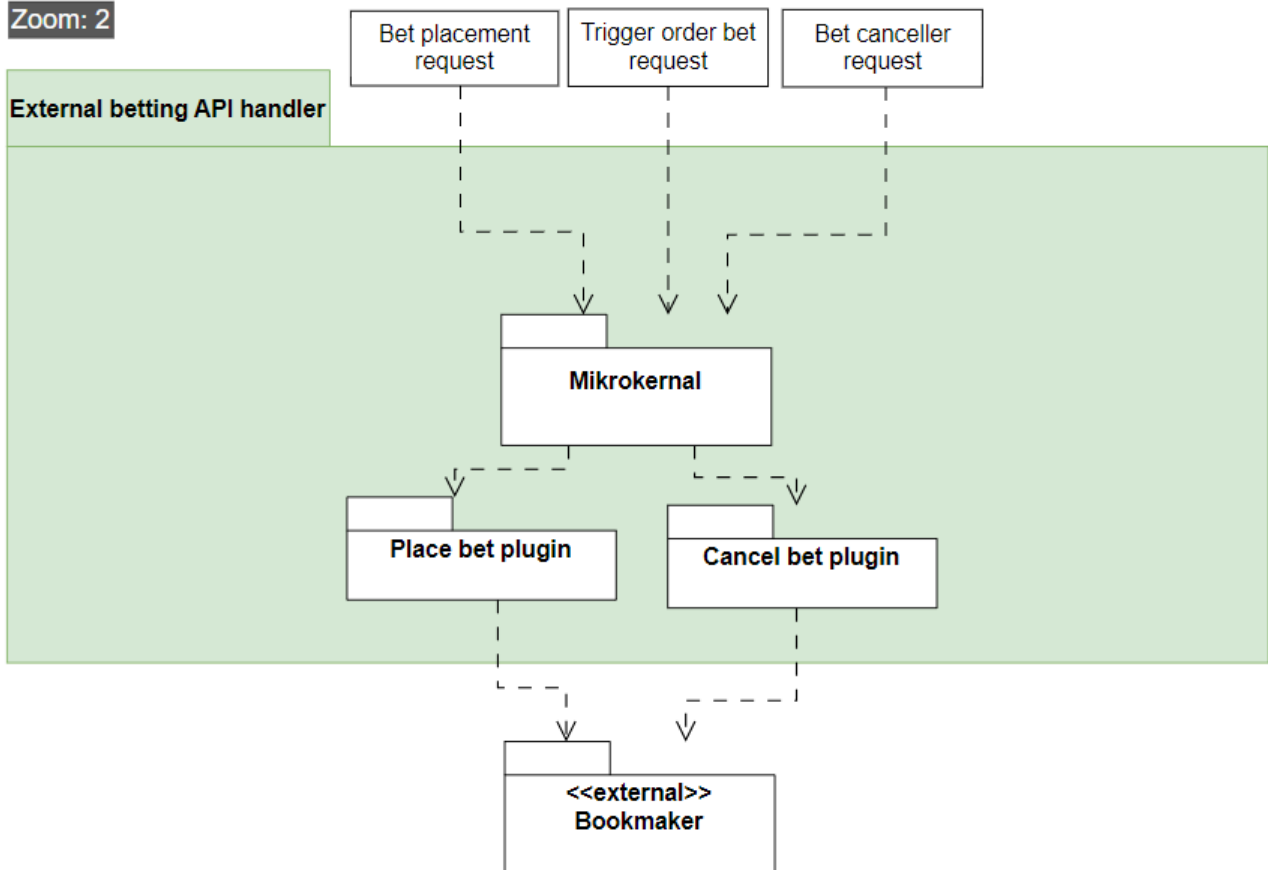


Figure 2: What happens inside the microkernel

Plugin actor model zoomed in view

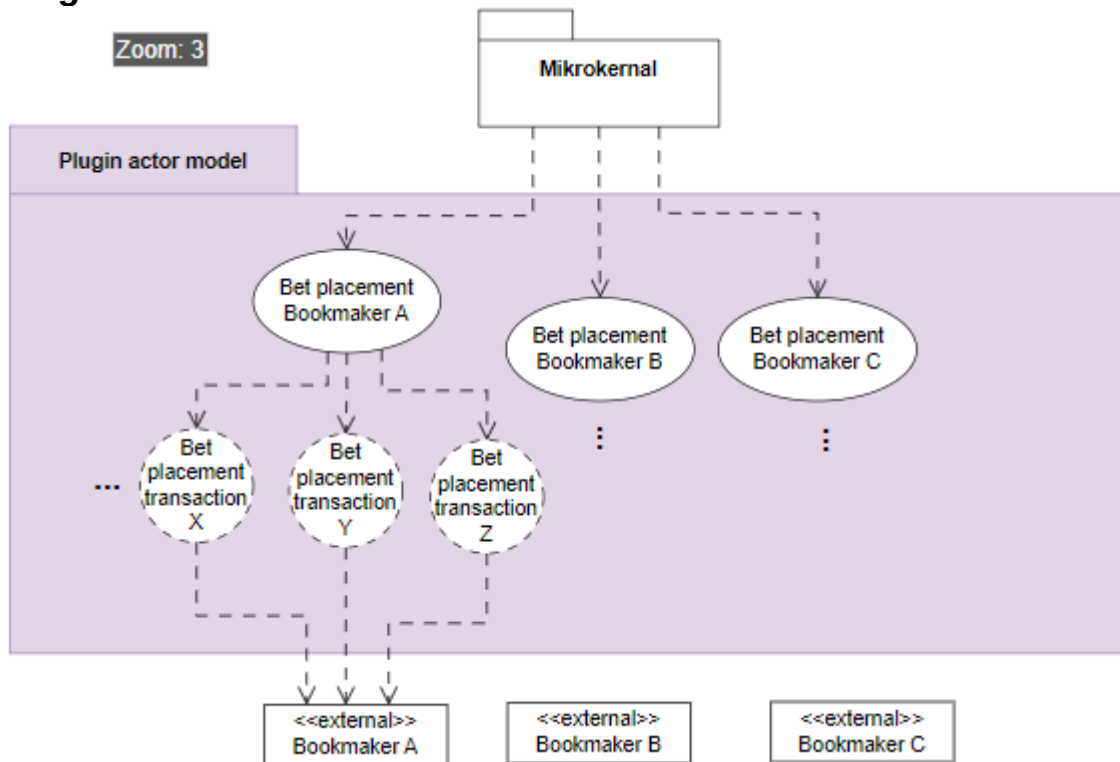


Figure 3: What happens inside the plugins that communicate with the bookmakers

Performance

Performance Patterns

A. Event broker [figure 1]

Pros

- The event broker can allow for low latency between odds being changed and the odds being displayed to our user. Each channel in the broker will be a specific match ID. Trigger orders will then act as subscribers on specific channels, using specific criteria being the odds and money placement.

Cons

- As the number of channels and subscribers increases, the broker can struggle to efficiently manage all connections, especially if channels are highly active. This can reduce performance.
- Traffic might be heavily unbalanced, as one channel might represent a local sports-match with 4 people in the audience and another represent the grand final match watched by millions. This can reduce performance.
- It can be difficult to ensure a correct ordering. In case we end up deciding that bigger bets should be prioritized higher than small bets. This can reduce usability.
- The broker might be the bottleneck, and a single point of failure. This can reduce availability.

B. Message Queuing using the Actor model [figure 3]

Pros

- Spinning up an actor to handle each transaction ensures greater resource utilization as they can work in parallel.
- Decoupling from other bookmaker, as each actor is designed to only communicate with a specific bookmaker. As thus, if the bookmaker changes the API, only a specific actor implementation is affected.
- This will also enhance safety qualities. Each actor will function as a “bet transaction”, handling cases where the bookmaker reject the bet due to odds changing, and then re-send the bet is within the users threshold setting for odds changing during placement.
- The actor model is suitable for horizontal scaling. Example by adding more “LeoVegas bookmaker actors”, in case there is high traffic on bet placements for LeoVegas.

Cons

- Spinning up actors introduces some time overhead.
- Inter-actor communication can introduce additional latency, thus reducing performance.

Performance Tactics

Control Resource Demand

Techniques: Prioritize events:

Pros:

- Events like trigger orders working with higher sums of real money should be prioritized compared to someone betting with fun-money or placing a small bet below 100 kr. This can be achieved with the event broker having a prioritization list. This will improve usability for the customer segment more important to us (The whales).

Cons:

- Less important events like updating the UI, gets higher latency.

Manage Resources

Techniques: Introduce concurrency:

Pros:

- Process multiple streams of betting data in parallel. This can be achieved with the actor model when handling bet placements. Each actor can act completely independent of other bookmaker actors, making their work “embarrassingly parallelizable”

Cons:

- Increased complexity in ensuring we don’t accidentally spin up two actors for the same task and places 2 bets etc.

Availability

Availability Patterns

A. Actor model [figure 3]

Pros

- The actor model can help preventing errors that causes the whole system to crash, by having the actors crash instead. Having a monitor on the actors lets the system know so it can notify the user or try again.

Cons

- Lots of communication handling is needed, which may introduce computation and complexity overhead.

Availability Tactics

Detect Faults

Techniques: Heartbeat:

Pros

- Regularly check the availability of external betting providers. Improves usability, if we can up-front tell a user that they can't place a bet at LeoVegas right now, compared to them trying to place a bet and afterwards getting told they can't do it.

Cons:

- Increases network traffic and monitoring overhead.

Recover from Faults

Techniques: Graceful degradation:

Pros

- When one provider fails, continue serving data from other providers. Achieved with actor model.

Cons

- Users may encounter reduced functionality in terms of bet placement options, which can lead to confusion or dissatisfaction.

Modifiability

Modifiability Patterns

A. Microkernel [figure 2]

Pros

- Provides a handler that supports placing/canceling bets on various external bookmaker API's. This way if new bookmakers appear or change, we only have to handle the change in the microkernel logic while the rest of the system remains the same. Thus improving modifiability by decoupling different bookmakers, and isolating the system affected by external changes.

Cons

- Extra abstraction layers may lead to performance overhead.

Modifiability Tactics

Increase Cohesion & Reduce Coupling

Techniques: Split Module & Encapsulate External Calls

Pros

- Having different actors [figure 3] for different bookmakers increases cohesion and since they work independently they have low coupling.
- Letting the event broker [figure 1] encapsulate by handle detecting odds. changes

Cons:

- Split module may decrease performance, because it adds more steps to the data-flow.
- Encapsulating in one place may introduce a bottleneck or single point of failure.

Safety

Safety Patterns

A. Actor model with message passing [figure 3]

(See performance section for more info). The actor handles the transaction process and makes sure it either succeeds or quickly detects if odds have changed, so the user can be notified.

Safety Tactics

Techniques: Exception Detection, Exception Handling & Input validation:

Pros:

- Monitoring the actors to ensure crashes are handled, will avoid major faults.
- Immediately reject and rollback transactions if anomalies are detected.
- Making sure invalid bets cannot be placed by having the actors closely communicate with the bookmaker API.

Cons:

1. Additional processing may introduce minor delays.
2. Complex exception logic can decrease maintainability, by making it harder to make changes will maintaining a growing exception handling system.

Architectural Prototyping

Architectural issue

When designing our prototype we would like to investigate how some of our identified important architectural qualities affects one another in an experimental manner. Specifically we would like to investigate how performance, modifiability and safety impact each other in the microkernel module. To facilitate this our prototype needs to be able to distribute bets in a way that simulates how the system would function in the real world. That is, we need to be able to handle uneven betting loads if e.g. a specific site has a good offer and gets disproportionately higher amount of bets than other bookmakers.

Designs

We will experiment with two different design variations: The actor model and a load balancer. We assume the former provides us with lower coupling between different actors and hereby ensuring better modifiability and safety. However, we want to explore the possible performance benefits of using a load balancer instead, and take a stand on whether the system is better suited with a load balancer or an actor model. The two designs are illustrated in Figure 4.

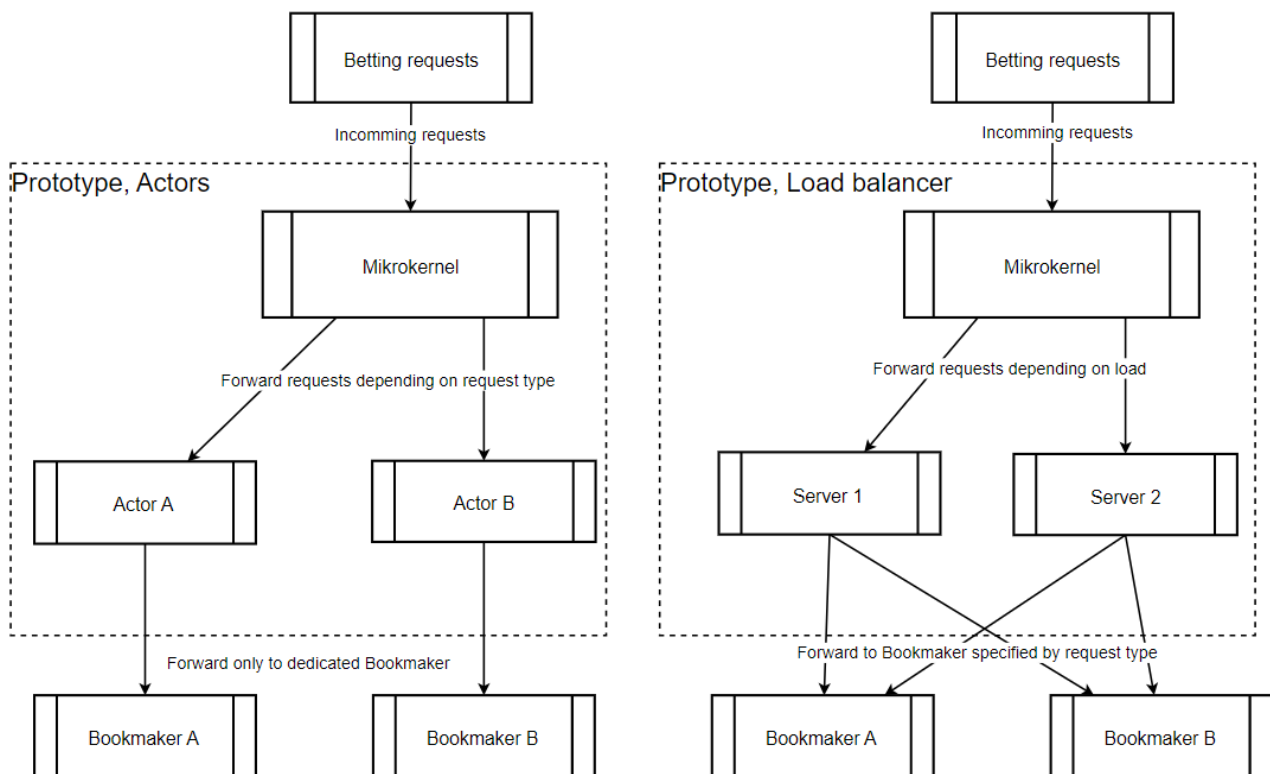


Figure 4: Actor Model and Load Balancer

In the following we will describe the two systems and highlight their pros and cons.

Actor model

In this design there is an actor for each bookmaker. The microkernel forwards betting requests to the actors based on the request type. If it is a Bet365 request, it will be forwarded to the actor that handles Bet365 requests etc.

Pro: Each actor is specialized to their given bookmaker. This ensures that the different logics/interfaces for different bookmakers are isolated, creating lower coupling and better modifiability. And if something goes wrong it only affects requests to the specific bookmaker, meaning we get higher safety.

Con: The actor handles all request for said bookmaker, so uneven load between bookmakers might overwork some actors and starve others leading to poor performance.

Load balancer

There are x (2 in our prototype, but more in real world) servers that be can distributed to. Each server can handle every kind of requests because the requests are distributed based on load. The least loaded server gets the next request.

Pro: With requests being distributed based on load instead of request type, we ensure even workload among the servers. Furthermore, it is easy to scale out horizontally by adding more servers, thereby always ensuring high performance.

Con: Each server must be able to handle requests for each bookmaker. This increases complexity, which can lead to errors and make it harder to modify when new bookmakers appear.

Prototyping setup and implementation

The code for our prototype can be found at:

<https://github.com/Grumblebob/Architecture/tree/main/Prototyping>

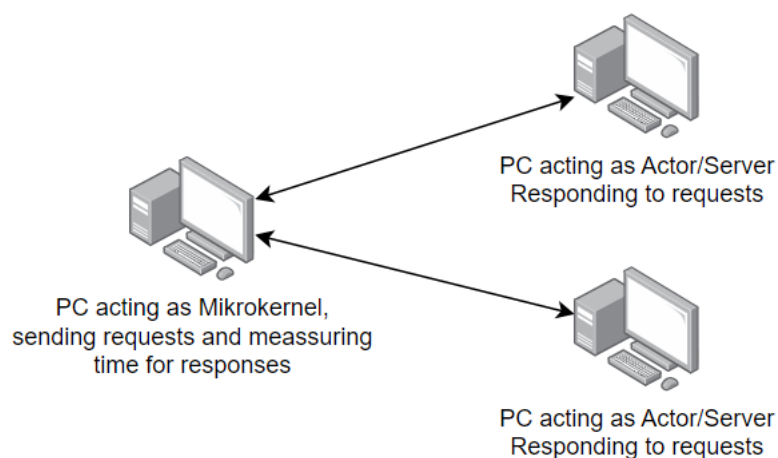


Figure 5: Our setup involves three PC's.

As seen in figure 5 we have one PC (A) acting as the microkernel. It simulates forwarding requests by sending requests with message parsing to two other PC's (B and C). The other PC's simply print to the console and respond to the requests, nothing else.

PC A has two implementations.

1. Actor model. PC A sends requests to B and C based on the type of the request. Type B goes to PC B and type C to PC C.
2. Load balancer. It keeps count of how many unanswered messages has been sent to each "server" and sends the next request to the one with fewest unanswered messages.

To simulate uneven workload we make 10 times more requests of type B than of type C. PC A performs the benchmarking based on how long it takes from requests are sent to responses are received. We used both a benchmarking library and our own implemented benchmarker.

Prototyping evaluation and conclusion

This is the printed result from the benchmark library (it was too slow with 10000 requests):

Method	TotalRequests	Mean	Error	StdDev	Allocated
-----	-----	-----: -----	-----: -----	-----: -----	-----: -----
RunActorBenchmark	100	254.1 ms	5.73 ms	16.53 ms	204.23 KB
RunActorBenchmark	1000	2,308.8 ms	44.52 ms	41.65 ms	2045.02 KB

Method	TotalRequests	Mean	Error	StdDev	Allocated
-----	-----	-----: -----	-----: -----	-----: -----	-----: -----
RunLoadBalancerBenchmark	100	248.9 ms	23.58 ms	66.13 ms	228.59 KB
RunLoadBalancerBenchmark	1000	2,098.3 ms	44.36 ms	128.70 ms	2287.53 KB

This is the results from our own benchmark implementation:

Approach	Requests	Fastest (ms)	Slowest (ms)	Average (ms)	Q1 (ms)	Median (ms)	Q3 (ms)
Actor	100	17,11	263,57	148,11	90,29	152,06	212,26
LoadBalancer	100	4,76	122,87	66,18	37,20	67,72	94,95
Actor	1000	2,62	2278,65	1055,93	423,49	1031,34	1694,64
LoadBalancer	1000	2,28	1273,27	646,01	321,39	663,64	970,55
Actor	10000	3,25	26505,02	11940,40	4215,14	11515,58	19142,97
LoadBalancer	10000	9,37	13584,77	7132,18	4032,39	7340,09	10421,72

As expected the load balancer is faster, giving us a performance gain of roughly 45% on average. This is a significant performance improvement to the actor model. Of cause these numbers may look different in the real world, but the question is whether the gain outweighs the potential costs of choosing this solution.

In our Architectural Analysis we identified performance as the most critical quality of our system with a score 14 from our stakeholder role-play. While modifiability and safety also hold importance with scores of 11 and 10 respectively, we think the clear priority on performance justifies the decision to chose the load balancer for our system. This approach aligns with our primary system goal of high performance, however it being at the cost of both modifiability and safety. This necessitates future work to make sure we still maintain a safe and modifiable system while maintaining high performance.

Architectural Evaluation (aSQA Method)

For the final evaluation of BestBets' architecture, we chose the **aSQA** method. The aSQA technique is a lightweight, metrics-driven approach for continuous architecture evaluation. It is well-suited to our agile, iterative process, with constant new lectures, feedback and learning - because it quantifies quality attribute levels and identifies focus areas with relatively little overhead. By contrast, **ATAM** is a scenario-based method that, while thorough, is resource-intensive and yields qualitative results (utility trees and risk themes) that can be harder to directly compare or prioritize.

The steps of aSQA

Metrics and levels

We will evaluate three of the key architectural quality attributes that we identified in the beginning of the project. Performance, Modifiability, Safety. These were also the ones we used for prototyping which makes it more appropriate to judge. We will use the scenarios we have defined for each, to base the scores off of.

We follow the same levels (1-5) as they used in the paper (Christensen et al.).

- 1: Unacceptable.
- 3: Acceptable.
- 5: Excellent.

Components

We will make an evaluation of some of the important components shown in diagrams in deliverable 2, our prototypes in deliverable 3 as well as an evaluation of our exploratory result of deliverable 3, which consists of combining our two prototypes into a single combined component.

In short our focus is on the following components:

First aSQA iteration:

- Deliverable 2:
 - Event broker, which is notified by the odds updater endpoint that external bookmakers uses, and then distributes to relevant trigger order consumers
- Deliverable 3:
 - Actor model & Actor plugin (Microkernel)
 - Load balancer

Second aSQA iteration:

- Exploratory result of deliverable 3:
 - Combination of actor model and load balancer.

Evaluation

We made our tables such that each shows the scores for a quality attribute. We did this to help ourselves by focusing on one attribute at a time instead of a component – This is mostly for our own sake, to make it easier to convert to a power point later.

Special notice

Actor model, load balancer and Combo components are three different solutions to the same task. Therefore they share the same target and importance.

Only the Actor model and Load balancer were actually implemented. The rest are hypothetically judged, this is shown with parenthesis.

Performance

Component	Target (T)	Current (C)	Health (H)	Importance (I)	Focus (F)
Event Broker	(3)	(1)	(3)	(2)	(2)
Actor model (plugin)	5	2	2	5	4
Load Balancer	5	5	5	5	1
Combo (Second aSQA iteration)	(5)	(3)	(3)	(5)	(3)

The event broker mostly affects trigger-orders, which is not a key money driver of this project. Hence the low importance.

For the odds updater we have not yet made a decision regarding the database that it updates, such as one single instance, or if we have multiple replicas with some being read-only for the sports overview. Therefore it has a low current score. It also has a high importance because it is prone to become a bottleneck for our data-flow.

Note how the actor model had high focus. The combo solution is an improvement and therefore after refinement the focus has dropped.

Modifiability

Component	Target (T)	Current (C)	Health (H)	Importance (I)	Focus (F)
Event Broker	(2)	(1)	(4)	(3)	(2)
Actor model (plugin)	4	2	3	4	3
Load Balancer	4	2	3	4	3
Combo (Second aSQA iteration)	(4)	(1)	(2)	(4)	(4)

The Actor model, Load balancer and Combo all lack implementations for different bookmakers. They only understand one interface for now. Hence the low current score and high importance. Combo is even lower due to its increased complexity.

Safety

Component	Target (T)	Current (C)	Health (H)	Importance (I)	Focus (F)
Event Broker	(2)	(1)	(4)	(2)	(1)
Actor model (plugin)	4	1	2	5	4
Load Balancer	4	1	2	5	4
Combo (Second aSQA iteration)	(4)	(1)	(2)	(5)	(4)

In general the event broker and odds updater only handle odds data and work internally. Failure will not have catastrophic consequences for the user. Therefore a low target.

The Actor model, Load balancer and Combo all lack implementations for retrying when performing a bet transaction with a bookmaker. Hence the low current score and high focus.

Overall evaluation

In general the event broker and the odds updater have low current scores because we have not worked much on them yet. The focus has so far been on the part of the system that Actor model, load balancer and combo can handle. Their importance scores are also higher because this is the part of the system where the key feature for our concept is implemented: Being able to bet on all bookmakers from one place. The exception to this is the performance for the odds updater. Since this is responsible for bringing in fresh data it must not become a bottle neck. So this also has a high focus.

Conclusion

Our current architectural design still needs a lot of work. Up until now our focus when prototyping has been on the “microkernel”. This is the part of the system that implements the key feature that lets users bet on different bookmakers from our system only. We started with a proposal of using an actor model, but after testing it against a load balancer we realized it needed to be optimized for better performance. After we changed from the actor model to a combo solution that involves a load balancer the performance was less of an issue. As seen in our evaluation there is still safety issues as we have not looked at retry logic when placing bets. This leaves the combo solution with a high focus for future safety improvements.

The odds updater and event broker, have more simple tasks of transferring/updating/reacting to data. Their focus scores were lower with the exception of the odds updater's performance. One of our requirements are to have "fresh" odds. This means the odds updater must not become a bottle neck on our system.

We also have left to decide how the database should work. Meaning if it should be a single instance or if we want replicas with read-only data.

Our architectural design has the overall functionality and flow, but lacks the finer details for certain parts.

Architecture backlog

3. Maintain and report on an architecture backlog. The backlog should contain decisions you have to make, work you need to do, issues you have etc. all in the scope of the system and not the project.

Our guide to our logs

1. Always remember to set dates.
2. Some idea might fit multiple logs/headers. It is not super important which is chosen, what is important is that it is written down.
3. Avoid merge conflicts of this Libre Office file by:
 - If alone, write it down in your own local file first, then meet with group to get it merged.
 - If with group, ensure only 1 is changing the log, then commit updates and notify group, so no concurrency issues arise.
4. If the idea is heavily conflicted by the group. Still write it down, but note down the disagreement.
5. Colors:
 - **Green:** Finished / implemented / done
 - **black:** Being worked on / being discussed
 - **Red:** Turned out to be irrelevant / discarded

Ideas related to business

10.02.2025

Should we be a bookmaker, or only an aggregator?

- If we are an aggregator showing the "best odds", it would be tough for us to take a margin, as that margin would possibly reduce the odds such that it isn't no longer the best odds, only due to our cut.
- Alternatively, we could do something like a subscriber model for our product. But i suspect that people would quickly calculate that as a percentage cut of their bettings as well, and therefore again reduce the chance that the customer would perceive it as the best odds available. Therefore, we think it would be easier to take a cut, on a service usually not available on other sides, which is acting as a stock broker just for betting. Meaning that people themselves would be able to open a position on some bet, and therefore act as the bookmaker themselves, for other people to buy their odds. This would cost a fee, quite similar to the ordinary fees one pays on the stock market.

Conclusion / What happened (04.03.2025): We are mostly an aggregator. We don't act as our own bookmaker, BUT we do allow our customers to act as individual bookmakers through our site, known as "community bets".

Does this concept give a total increase in new customers across betting sites, or does it eat from the existing customer base.

- So one of the biggest podcasts out there from Conan O' Brian, who is mainly on Spotify with audio only. They are now experimenting with putting episodes on Youtube with video as well. But they are afraid that it won't give more customers, instead people will migrate from Spotify to Youtube. This is a concern for the betting vendors as well, where there are mainly two scenarios:
 - Best case: Some customer who only uses X, instead now uses our platform and therefore indirectly ends up using A, B, C, D.... who never had a chance before to be used by this customer. They all see an increase, because this customer who now have more options and advantages starts betting 250 % more. So even though site X will see a decrease from this customer, there might be other customers who only uses A, that will now also start using X.
 - Worst case: Some customer who only bets 50 kr. each month on X, will now still only place 50 kr each month, but now on A, B, X. Thus no additional revenue would be added to the pot.

Should our concept only serve sport?

Most of these sites also have:

- Bingo
- Casino & Live casino
- Poker & other card games
- Raffles
- eSport (One might claim they are the same)
- Quizzes and giveaways

While these might make sense in our business scope, it will be too much for this 1 year project, therefore we only focus on sport - not because it is the best choice, but it is feasible within our scope of this course.

But alternatively, we can implement casino easily through casino providers. So ALL of the betting sites we checked with danish license (42), all used external casino vendors, such as pragmatic.com. As such, we can easily and efficiently have a casino page as well, using external vendors. There is no "down-payment", as we act merely as affiliate, and get a provision from the vendors. This might be of conflict for our own betting site key partners, as if they start using our site, the potential income generated on slot-machines, will be reduced from our partners, as we eventually eat their customers. Alternatively is we don't provide casino (slots, live roulette etc.), as to please our key partners.

Conclusion / What happened (17.02.2025): We limit ourselves to sports bets for this project.

Special offers & bonuses

Is it feasible for our site to have offers, such as a welcome bonus?

This might be rough, because that would give the customers multiple stacked welcome bonuses, one from our site, as well as one from the site they are placing the bet on.

Or they might have to be designed in a smart way, we currently can't think of.

Follow the best users, opt-in, always on, or never show?

Should we allow people to see the most successful users so they can mirror their bets? If so, should this feature always be on, or something the user opts for.

We must show betting limitations

The UI must show the limitations of bets. "This bet gives odds 1.8 but max bet amount is 75kr."

Legal concerns

There must be compliance of "Rofus" and "Stop Spillet". Furthermore, there must be a timer that makes the user aware of their time spent betting. All of these and additional requirements from Spillemyndighederne must be implemented.

Separate withdraw-able cash and bonus

Bonus money staying in the betting site acts differently than cash that can be withdrawn directly. The site must display how much is bonus and how much is not.

Bonuses must remain on the bookmaker, and can't be transferred to our site.

Forum?

Having a forum on the site will act as marketing when users encourage each other to place bets.

Streaming?

We could use streaming through for example bet365 so users who want to watch matches live can be satisfied, without having to invest in our own streaming setup with people filming the match.

BetBørs is apparently already a thing

Our “community bets” idea already exist on BetFair. We thought it was a unique idea. Even though it already exists, we are going to continue with the idea, as we think the idea is great for showcasing various software architecture concepts.

International?

Could this be expanded to be used outside of Denmark? Could this site easily be operated in USA as well?

Two customer profiles

We actually have at least two customer segments.

- Users who want to bet.
- Bookmakers. Imagine a newly created bookmaker, who wants to make a splash and quickly attract new users. They could sign up through us, and “out-bid” other vendors with odds, to quickly reach a massive audience.

EU wallet

Should this site serve EU customers, through the newly EU created product “EU Wallet” – This means that we should alternatively mark our diagrams as “MitID / EU Wallet”.

Where is the money

Having them place money on our site makes it easier for the user to just put 200 and then bet where ever, instead of having to place 200 on multiple site. The down side is that we now need to handle security and all the behind the scenes transactions. But this idea proposal, is one of the key differences between just “googling the best odds” and having a central betting platform, taking care of various quirks, such as minimal down-payment for the customers.

Conclusion / What happened (04.03.2025): We included a bank to our architecture so we can hold the money, this way the user does not need to deposit money on every site.

Cancel bets

Users should be able to “cash out”, cancel their bet, etc.

This system should work very similar to placing bets, such as getting rejected and prompted in case they are trying to cash-out and the odds have changed during the process.

Conclusion / What happened (04.03.2025): The microkernel can forward cancel requests as well as bet placement requests.

Ideas related to patterns

10.02.2025

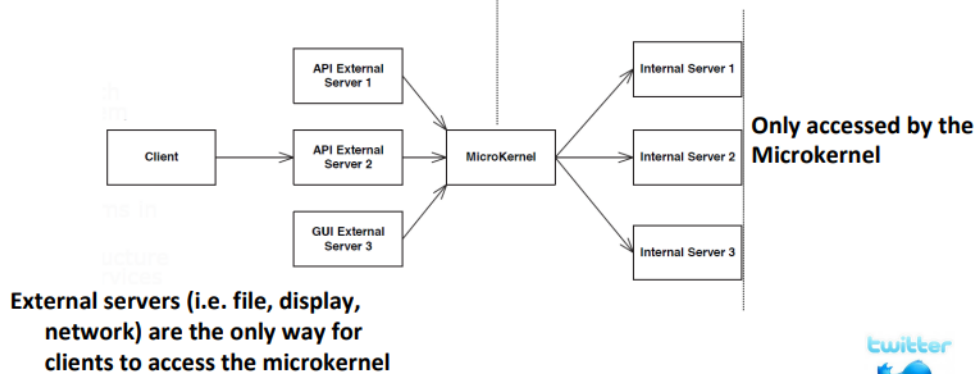
Pattern for registering many different vendors through our site.

We need a system that handles our external bookmakers different processes, as to minimize the pain of the user, needing to create a profile on multiple sites.

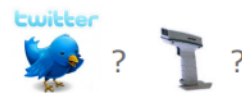
Pattern: Microkernel

Problem: system family in which different versions of a system need to be supported

Realizes services that all systems in the family need.
Plug-and-play infra for system-specific services



Canonical example? Microkernel OS



IT UNIVERSITY OF COPENHAGEN

29

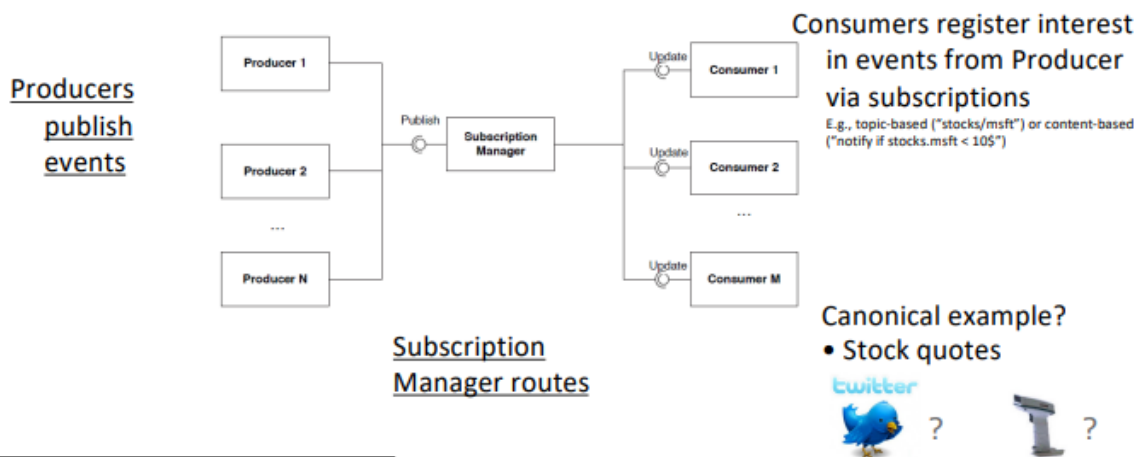
Conclusion / What happened (04.03.2025): Implemented in our diagrams.

For trigger order like a stock broker, when odds reach $X < Y$

If a user wants to place a bet on a match reaching above 2.0 in odds, it needs to be done quickly when the odds change. Thus the customer should “subscribe” to a match, where the “publisher” is the odds. If odds reach a certain threshold, the subscriber pipeline should automatically place the relevant bets.

Publish/Subscribe (Component Interaction)

Problem: Event consumers and producers should be decoupled. Many consumers should receive events from one producer



IT UNIVERSITY OF COPENHAGEN

27

Alternatively, we want to also look at:

Event Broker

Problem: how do we update a consumer about a state change while keeping them decoupled from the producers ?

Event-Driven Architecture approach

Consumers “subscribe” to events and receive notifications when they occur.

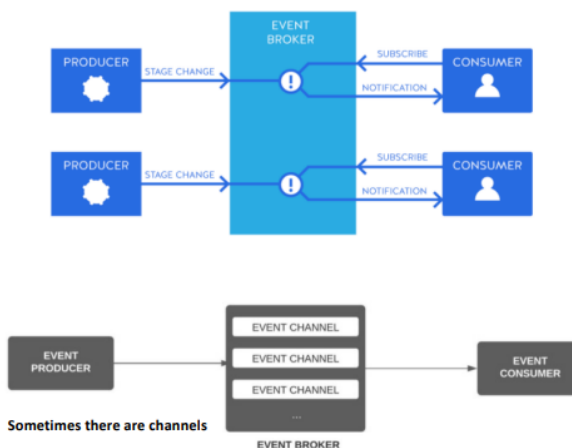
Producer is not aware of the consumer who receives it

The event **broker** in the middle

- allows both parties to scale and evolve in a loosely coupled manner
- might store events or not

Challenges: Broker should be

- scalable
- high performant
- fault-tolerant



IT UNIVERSITY OF COPENHAGEN

<http://radar.oreilly.com/2015/02/variations-in-event-driven-architecture.html>

39

Conclusion / What happened (03.03.2025):

We decided to go with:

Trigger orders uses Event broker. The producer are “the changed odds on events”, and subscriber are the customers placing trigger orders.

This will then get forwarded to our Microkernel responsible for communicating with the different Actor models using message passing to attempt to place the bet for the customer and handling errors in case the odds have changed, by checking the odds-changed-treshold on the customer, and reattempts if within border, otherwise cancels – acting as a “transaction”.

03.03.2025

Should we have multiple copies of the database?

We could envision something like this:

1. Bookmakers update their odds.
2. The updates odds gets sent to our API endpoint, which updates our odds database.
3. The odds database could have a replicated read-only database.
4. Our sports-overview uses only the read-only version.

If so, what do we actually gain from this – we need to prototype this.

We need to show a diagram for what happens AFTER a placement is made

Such as updating account information, sending confirmations, etc.

05.03.2025

Compare actor model against alternatives

Experimental prototyping will reveal more about whether the overhead of actors will be worth the benefits.

Conclusion / What happened (10.03.2025): See section about architectural prototyping.

Ideas related to tactics

10.02.2025

Availability

We need to monitor the health of a lot of different external sites, and should use tactics such as “heartbeat”

Conclusion / What happened (04.03.2025): Added in the tactics section for deliverable 2.

03.03.2025

Should we prioritize certain bets?

Such as, should a bet placement of 1,1 million be prioritized above a freebet of 50 kr.

If so we might want to look at: Tactic: Performance: Control Resource demand: Prioritize events.

Conclusion / What happened (04.03.2025): Added in the tactics section for deliverable 2.

Confirmed Decisions

10.02.2025

Fictitious project

We have made some fictitious assumptions, to make this project interesting to work with.

- All danish bookmakers have agreed to be part of BestBets.
- All bookmakers already have some API for placing/canceling bets.
- There are no legal troubles, such as not complying with various laws from Spillemyndighederne
- Transfers can be made between various bookmakers, and our own bank, as to avoid the restriction of “transfers must be through a MitId provider og NemKonto”.

Discarded decisions

10.02.2025

Placing multiple bets where one fails

On ordinary betting sites users can place multiple bets “in a package”, where the site is able to reject the whole package if one bet fails (such as a rapid change in odds). Our system will not be able to handle this, because the package might contain different sites, and as such we can’t cancel a bet on site A, if a bet fails on site B. As such it would be easier to not support package bets on our site.

11.02.2025

API og Data-scraping

We are not going to do data-scraping. If we do data-scraping, it would be hard to actually place bets on the respective sites, without proper confirmations etc. It would also be insanely resource intensive, to data-scrape a bunch of sites and also expect data to be completely fresh for the user. To avoid complications of legal situations and latency, we are going to make an assumption, that all bookmakers are willing to participate and already have an API giving the necessary data.

Work/issues needing to be done

11.02.2025

MitID integration

As we are doing betting, we need a license from "Spillemyndighederne", and follow their related laws. That includes MitID integration, for logging in, and also for getting payments out. Should we spent a lot of time and illustrations showing this, or is it simply "given" by the case, and illustration is not really necessary??

17.02.2025

Diagram showing bounds of our own bank connection and various payment providers

It can be a bit confusing, when and where the user is going directly through our banking connection, and when the user is going through a payment provider such as Visa & MasterCard.

Such as, what is the trace of money, when a user:

1. Deposits money
2. Places a bet
 - The bet loses
 - The bet wins.
3. Withdraws money

Statistics

Most sites support statistics, such as RTP on a specific feature that last X days. We should show general statistics, such as RTP, win-rate, etc.

11.03.2025

Update all relevant diagrams after prototyping

Once we have settled on how our forwarding from the microkernel should be handled, we need to update the diagrams in section about synthesis and perhaps overview.

17.03.2025

How should the load balancer approach safety and modifiability concerns

If we choose to use a load balancer instead of an actor model we sacrifice modifiability and safety. How do we ensure that it doesn't become too much a problem. How should we handle errors and new bookmakers?