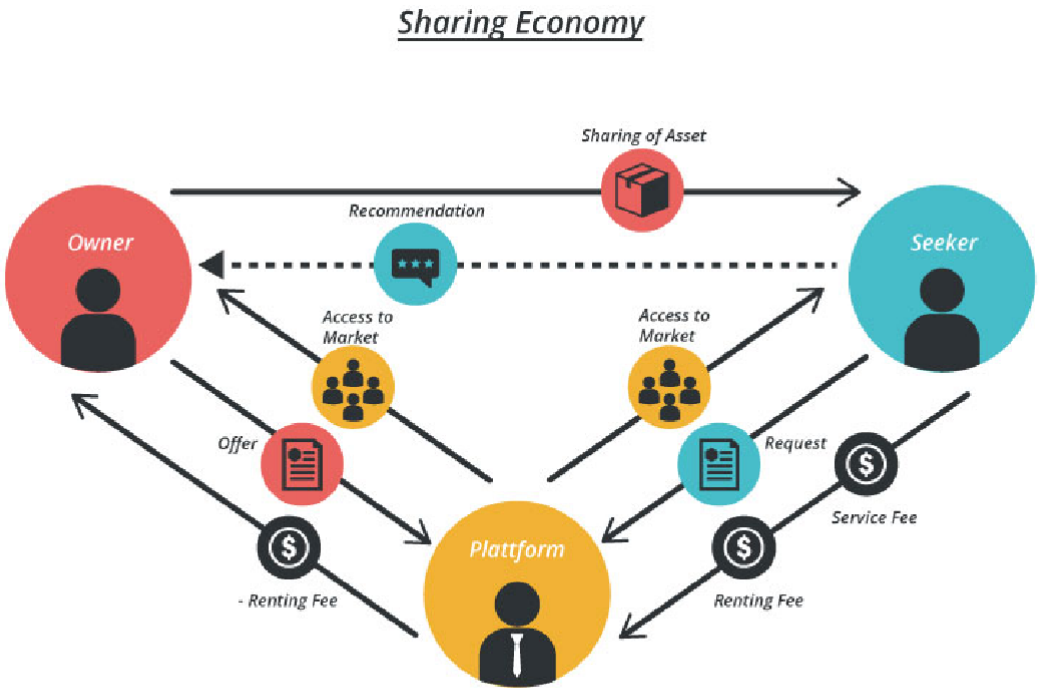
Marketplace Ideation

https://www.sharetribe.com/academy/how-to-come-up-with-a-great-marketplace-idea/



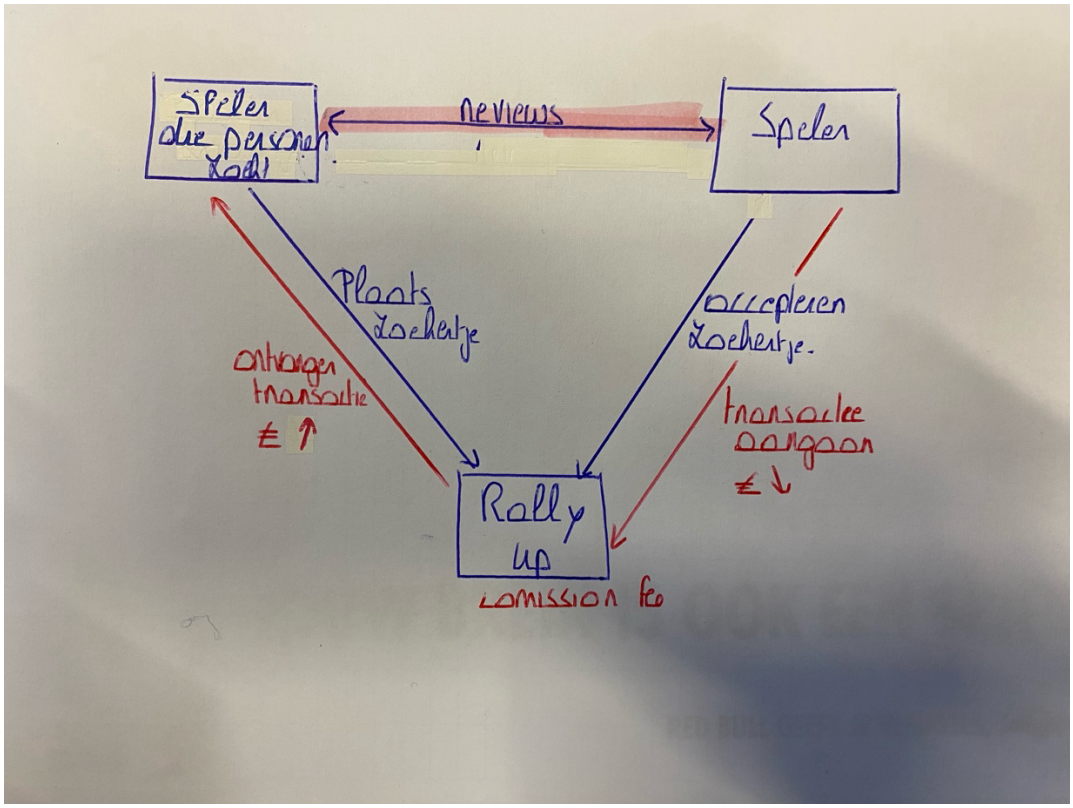
Platform name:

Rally Up

Short description:

Onze app, **"Rally Up"**, is ontworpen om padelspelers met elkaar in contact te brengen. Ontdek nieuwe spelers van verschillende niveaus, vind eenvoudig vervangers wanneer iemand onverwacht afzegt, of leer gewoon nieuwe mensen kennen op een sportieve en toegankelijke manier. Gebruikers kunnen elkaar beoordelen via reviews op basis van gedrag, speelstijl en niveau, waardoor je altijd weet met wie je speelt. **Rally Up: waar sport en vriendschap elkaar ontmoeten!**,

Model or idea sketch:


Speler die persoon zoekt
reviews
Speler
Plaats zoekertje
accepteren zoekertje
ontvangen transactie
€ ↑
transactie aangaan
€ ↓
Rally up
commission fee

Select Properties in the Taxonomy

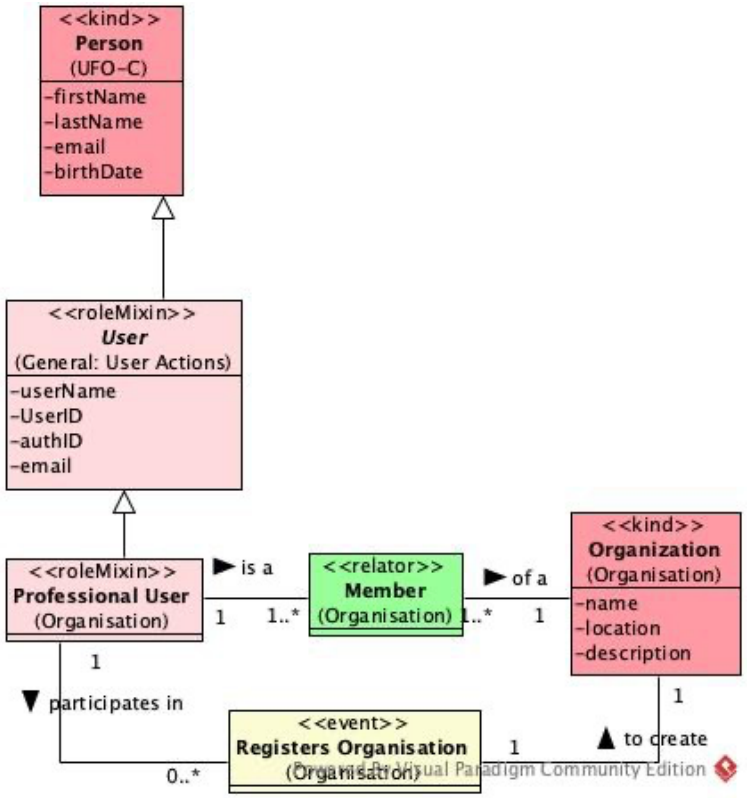
Dimension	Value			
User Type[m]	Person		Organization	
Listing Kind[m]	Good Transfer		Service	
Listing Type[m]	Physical Good[d]	Digital Good[d]	Offline Service[d]	Online Service[d]
Frequency			One-Time[e, d]	Recurring[e, d]
Quantity[m]	One[e]		Many[e]	
Price Discovery	Set by Provider[e]	Set by Customer[e]	Set by Market[e]	
~~*Price Calculation*~~	By Quantity[d]	By Feature[d]	Auction[e, d]	Quote[e, d]
~~*Conversation System*~~	Listing Conversation		Transaction Conversation	
Review by	By Customer		By Provider	
Review of	Of Listing[e, d]	Of Provider[e, d]	Of Customer	
~~*Trust and Safety*~~	ID Verification	Badges	In-app reporting	
Revenue Stream	Subscription	Commission	Fixed Fee	Listing Fee
Revenue Source	Customer		Provider[d]	

Mandatory: 'm', Exclusive: 'e', Dependent: 'd' and thick boxes, grey shading: no ontology modules (yet)

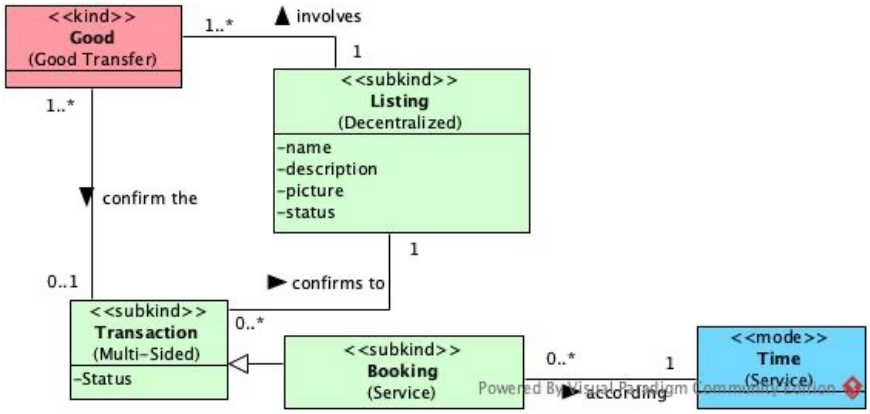
Develop Platform-Specific Ontology

1. Install Visual Paradigm Community Edition (CE): https://www.visual-paradigm.com/download/community.jsp
2. Download the ZIP file ontouml-vp-plugin on https://github.com/OntoUML/ontouml-vp-plugin/releases and install in Visual Paradigm CE via *Help > Install Plugin*
3. Download the Digital Platform Ontology on Github https://github.com/tdrave/Digital_Platform_Ontology and open the *Digital Platform Ontology (DPO+).vpp* file in Visual Paradigm CE
4. Drag and drop relevant classes in the ontology based on the properties selected in the taxonomy (with Thomas)
5. Ctrl+a, right click on a class, *presentation options => show owner => show name only*
6. Reorder, rename and /or delete classes and relationships
7. Add/delete attributes

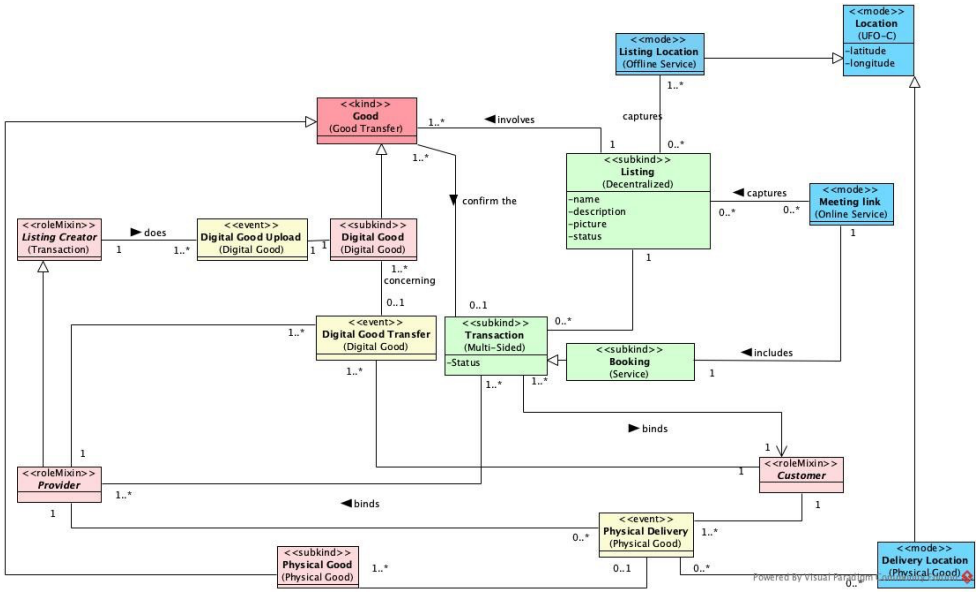
User Type^{m}	Person	Organization



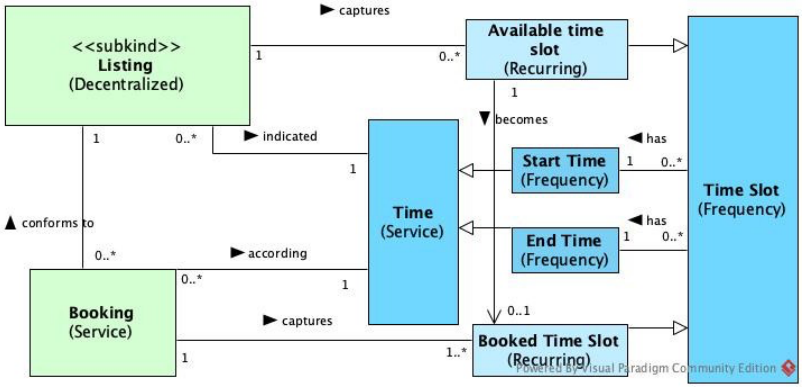
Listing Kind^{m}	Good Transfer	Service



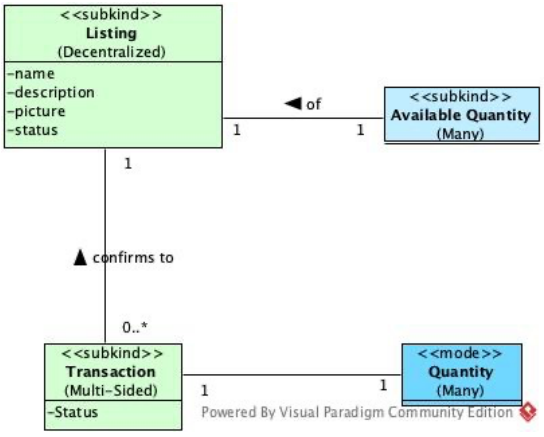
Listing Type[m]	Physical Good[d]	Digital Good[d]	Offline Service[d]	Online Service[d]



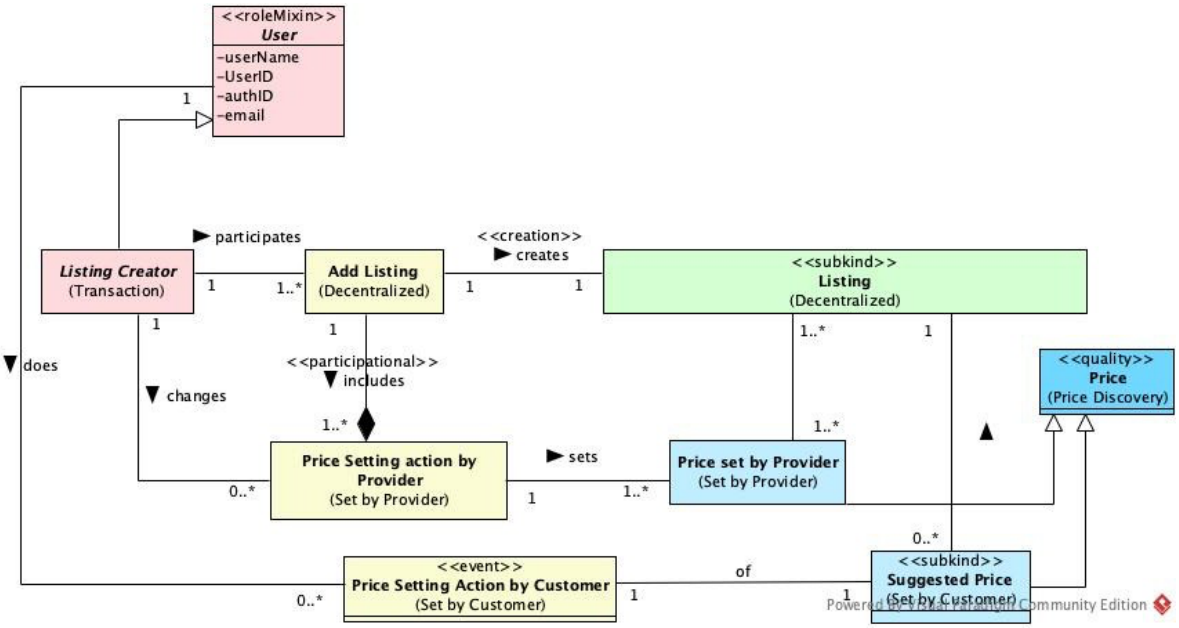
Frequency		One-Time[e, d]	Recurring[e, d]



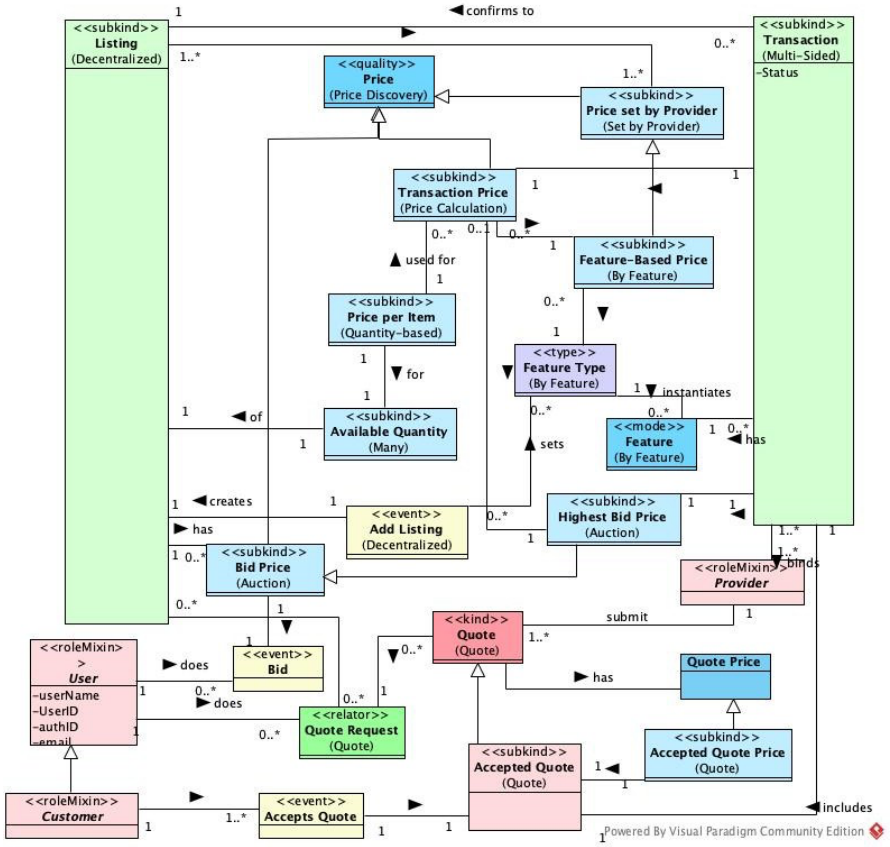
Quantity[m]	One[e]	Many[e]



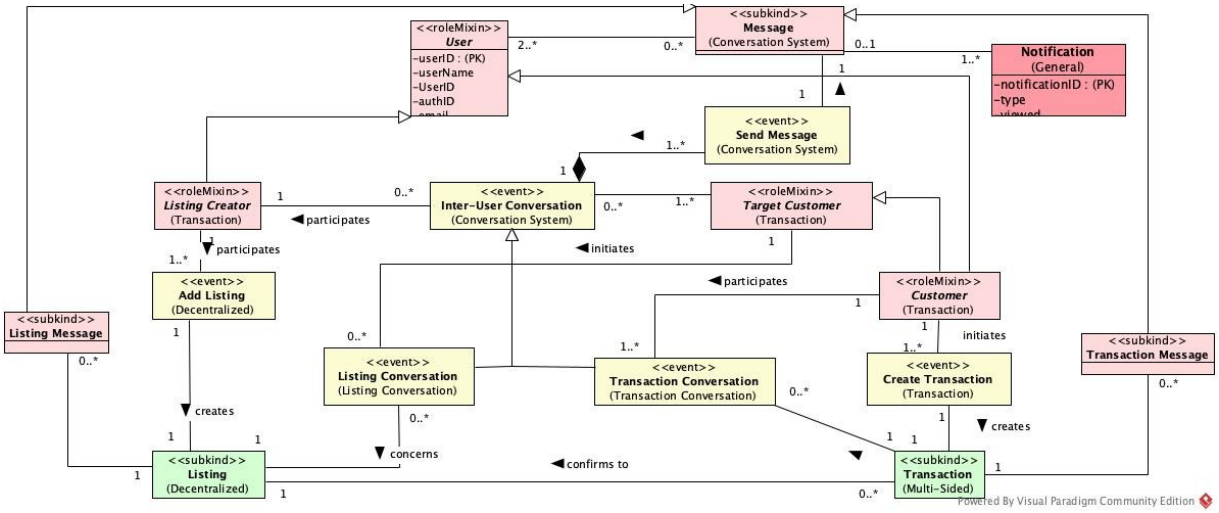
Price Discovery	Set by Provider[e]	Set by Customer[e]	Set by Market[e]



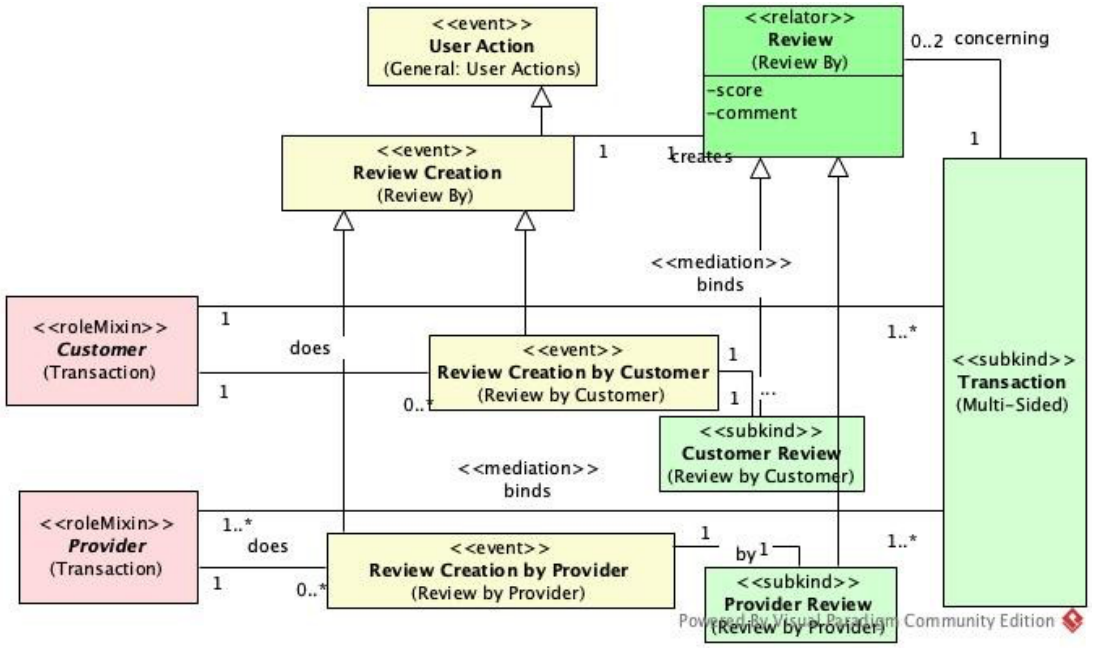
Price Calculation	By Quantity[d]	By Feature[d]	Auction[e, d]	Quote[e, d]



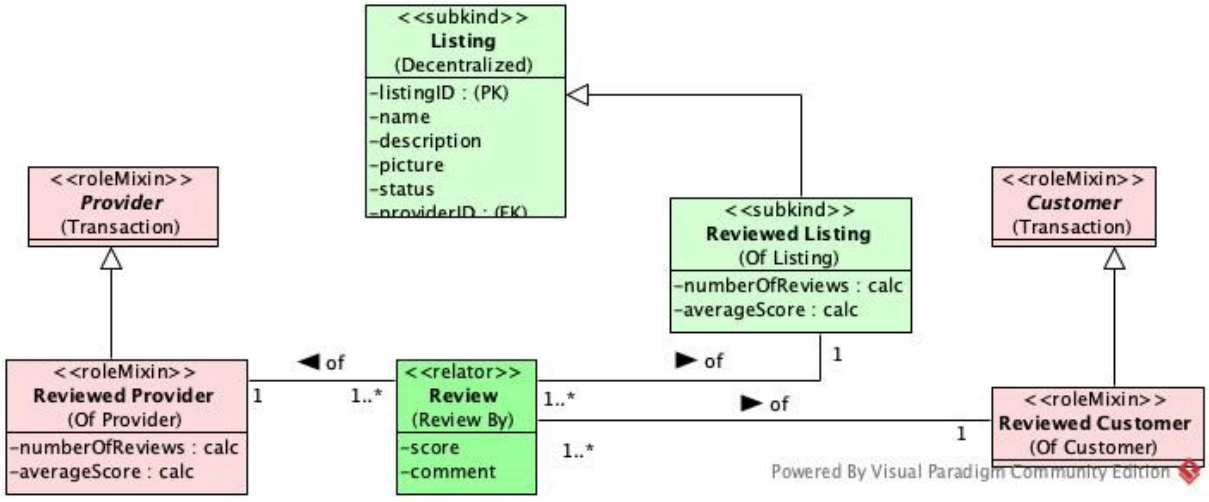
Conversation System	Listing Conversation	Transaction Conversation



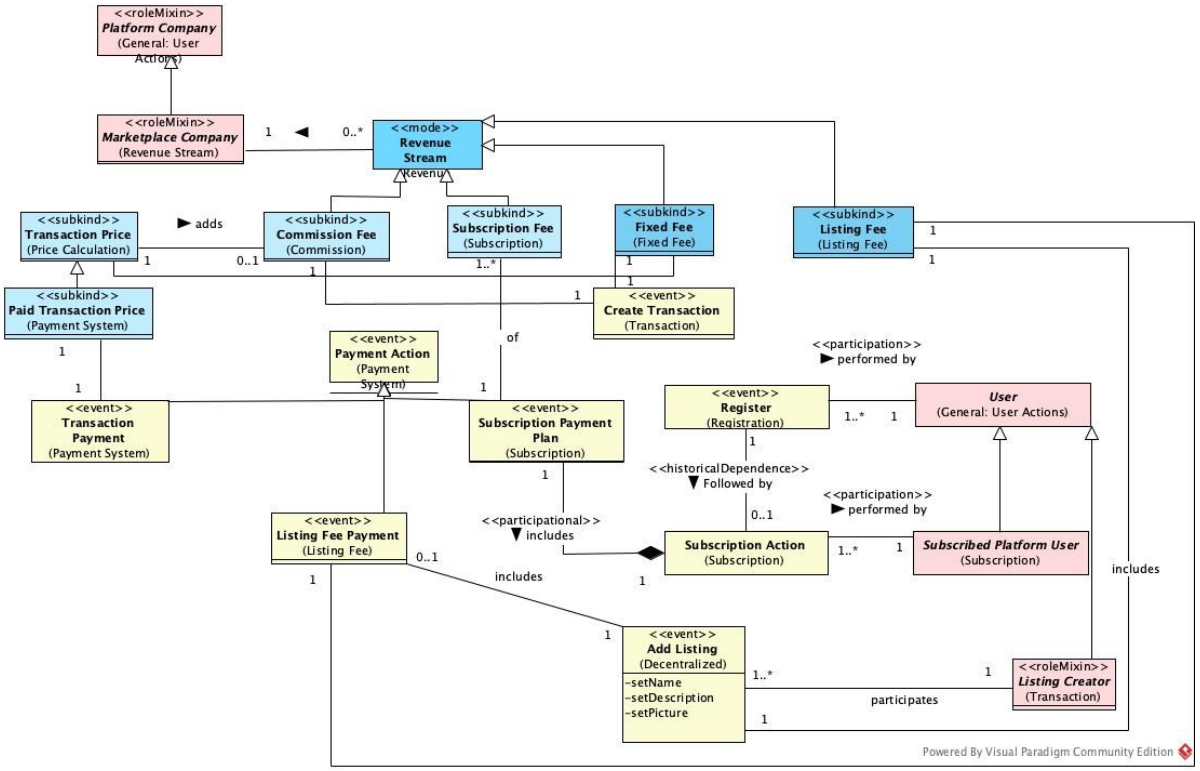
Review by	By Customer	By Provider



Review of	Of Listing[e, d]	Of Provider[e, d]	Of Customer



Revenue Stream	Subscription	Commission	Fixed Fee	Listing Fee



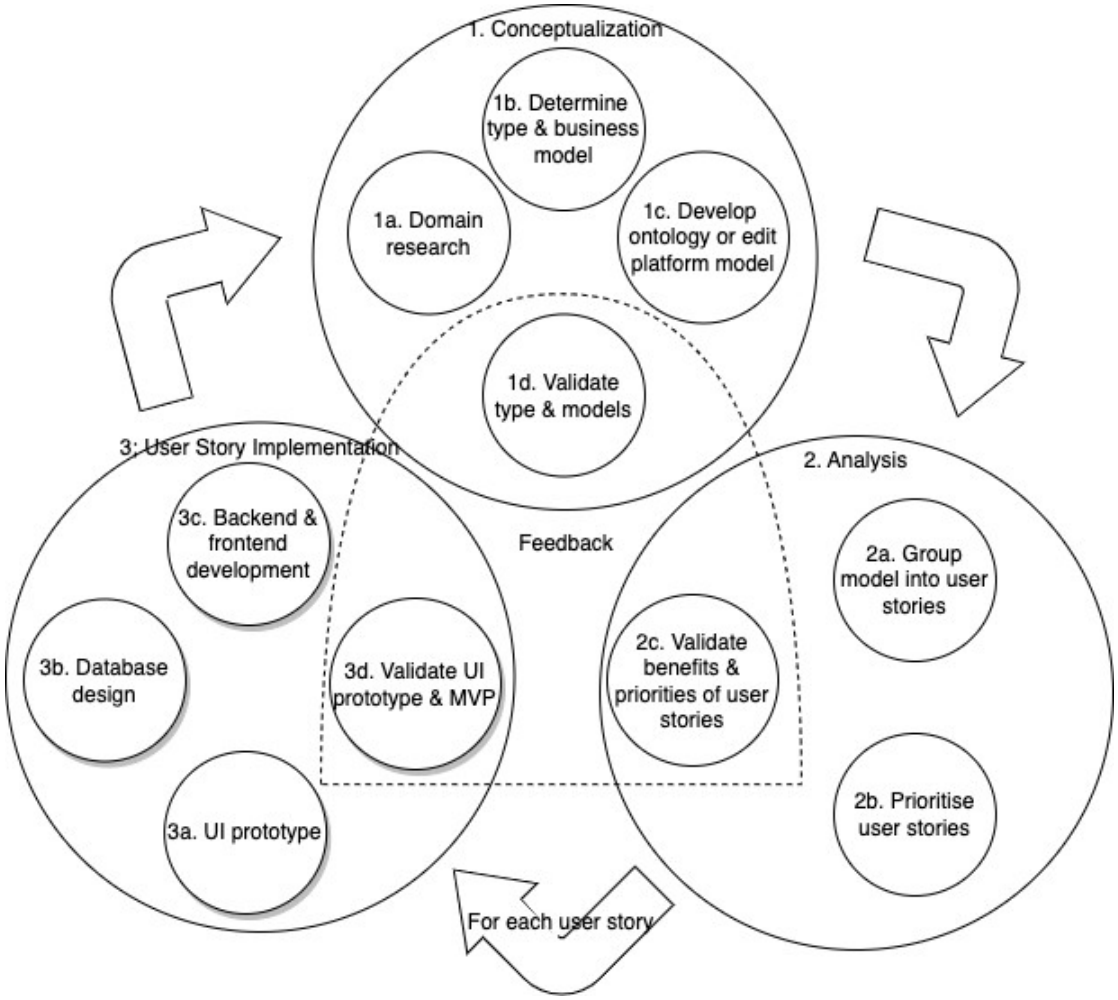
Write User Stories: hoogste prioriteit = 1

As a (role) *Red class*	**I want to (goal)** *yellow class*	**So that (benefit)**	**Prio**
Gebruiker	Beschikbare zoekertjes bekijken	Info heeft over openstaande zoekertjes	3
Platform bezoeker	Registreren	Om toegang te krijgen tot het platform	1
Gebruiker	Inloggen	Om toegang te krijgen tot profiel en beschikbare zoekertjes	2
Gebruiker	Zoekertjes kunnen toevoegen	Hij met gevraagd aantal personen kan gaan padellen	4
Gebruiker	Kan andere speler oordelen	Anderen gebruik kunnen profiteren van feedback	5
Gebruiker	Speelniveau bewerken	Zodat anderen het juiste niveau kunnen zien	8
Gebruiker	Kunnen filteren	Om snel en voor hem gepaste zoekertjes te vinden	7
Rally-up-eigenaars	Kunnen geld verdienen via commission fee	Zodat de app winstgevend is	6
Gebruiker	Wil meldingen ontvangen	Op de hoogte blijft van aanvaarde zoekertjes en rescensies	16
Rally-up-eigenaars	Willen data + statistieken van het platform bekijken	Ze trends en gebruikersgedrag kunnen analyseren	17
Gebruiker	Kan andere profielen opzoeken	Beter geïnfomeerd te kunnen zijn over andere users	10
Gebruiker	Kan zijn eigen profiel alsook zoekertje aanpassen	Persoonlijke informatie up-to-date kan gehouden worden	9

Gebruiker	Eigen reviews kunnen bekijken op mijn profiel	Ik beter geïnformeerd ben over mijn eigen speelgedrag	11
Gebruiker	Mijn ontavangende en betalende transacties bekijken	Ik kan zien welke van mijn zoekertjes geboekt zijn en de welke ik heb geboekt	13
Gebruiker	Mijn boekingen toevoegen aan mijn persoonlijke agenda	Ik gemakkelijk mijn afspraak kan terug vinden	14
Rally-up-eigenaars	Willen dat zoekertjes waarvan de datum werd overschreden automatisch uit ‘beschikbare zoekertjes’ worden gehaald	Er geen verwarring kan ontstaan	12
Gebruiker	Zoekertjes vinden die op intelligente wijze (meest gespeelde dag) speciaal voor mij worden voorgesteld	Gemakkelijk en accuraat een bijpassend zoekertje kan vinden	15
Gebruiker	Zoekertjes vinden die speciaal voor mij zijn via gedrag-analyserend algoritme	Nieuwe mensen kan leren kennen om mee te gaan padellen die goed bij de gebruiker passen	18

Afgewerkt: tot en met 15

Ontology-driven MVP development method

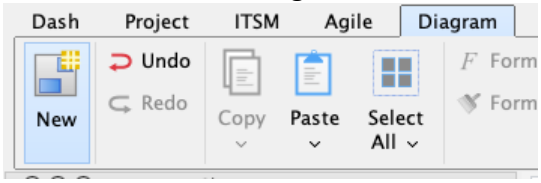


3a: UI Prototype

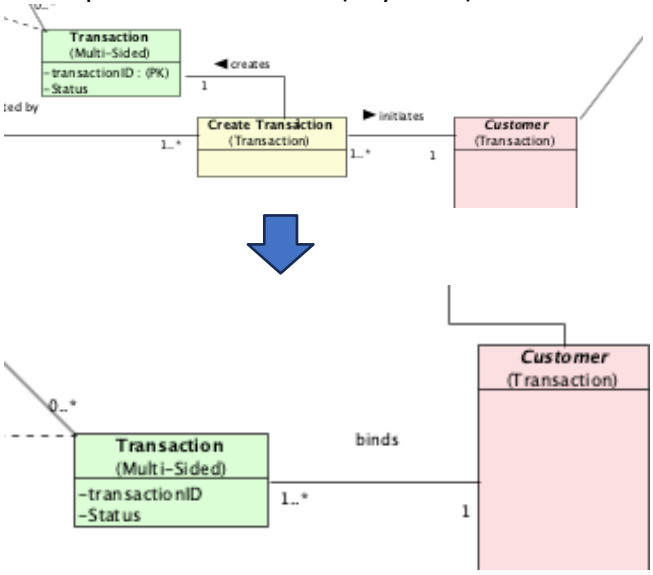
www.figma.com
https://help.figma.com/hc/en-us/categories/360002051613-Get-started
https://www.youtube.com/watch?v=HZuk6Wkx_Eg

3b: From ontology to Entity-Relationship Diagram (ERD): Database Design

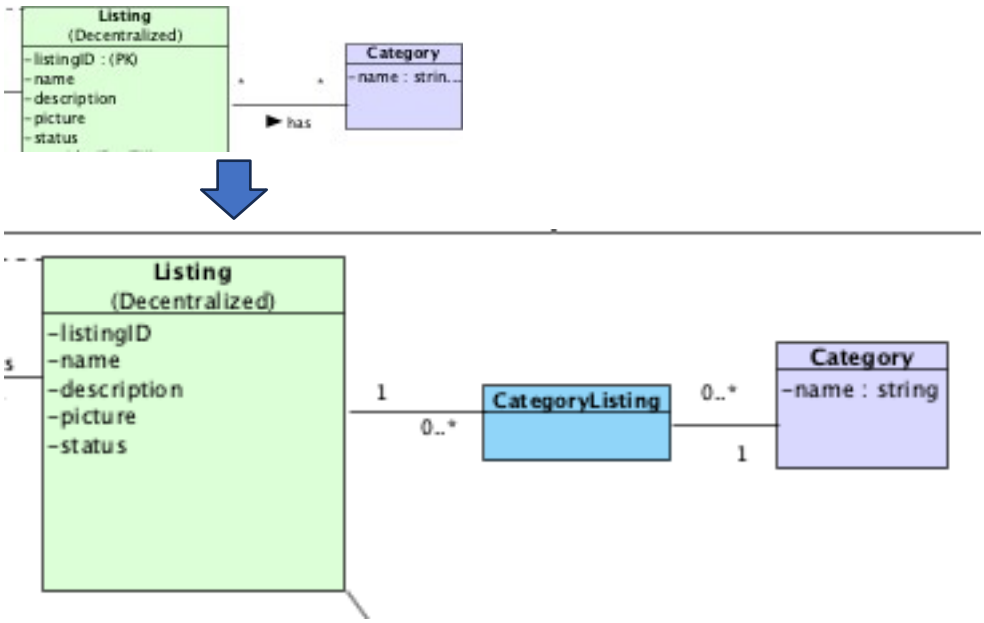
1. Create a new *class* diagram



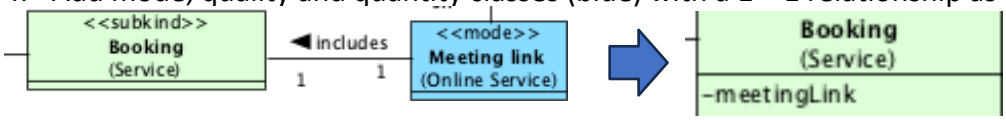
2. Replace event classes (in yellow) with relationships where needed



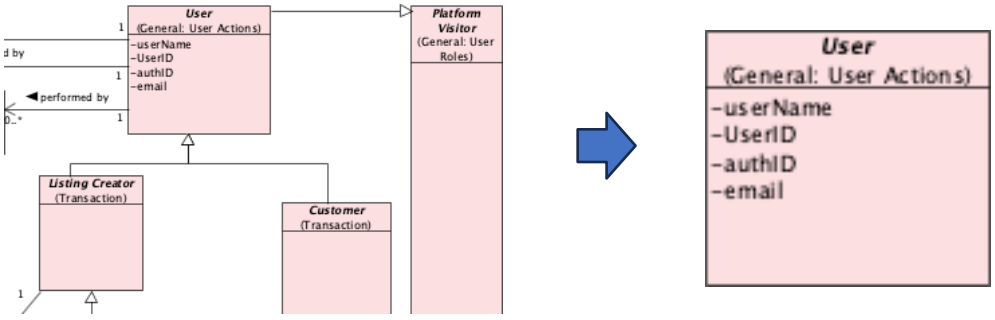
3. Introduce a new table between * to * relationship



4. Add mode, quality and quantity classes (blue) with a 1 – 1 relationship as attributes

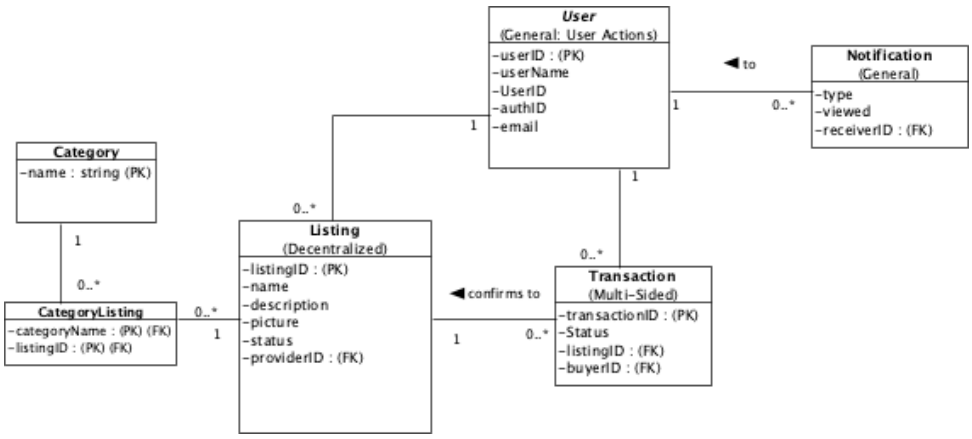


5. Solve each generalization-specialization relationship by changing super - subclasses into
    a. One table for **all** classes (both superclass and subclasses)
    b. One table for **each subclass** (not for the superclass)
    c. One table for **each** class (super and subclasses)



6. Delete enumerations
7. Add Primary Keys (One for each table) and foreign keys (One for each relationship)
8. Optional: Set background style to white (*Styles and Formatting*) and hide owner (*Presentation Options*)
9. Optional: Add datatypes (text, number, date)
10. Optional: Hide FKs in your ontology model (*Selection => Hide*)

Result:



3c: From Ontology to UI Prototype and web application: Backend & frontend development

These steps should be executed in parallel. Focus on the user stories with highest prioritization (Prio):

- Create a screen in your UI prototype (e.g., Figma) for each event (marked in yellow) in your ontology.
- In models.py, create a class for each entity in the Entity-Relationship Diagram (ERD)

```
class Listing(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    listing_name = db.Column(db.String(100), nullable=False)
    price = db.Column(db.Float, nullable=False)
    user_id = db.Column(db.Integer, db.ForeignKey('user.id'), nullable=False)
```

- For each relevant event (marked in yellow) in your ontology, create an HTML template based on the screen of you UI prototype

- In *routes.py,* create a route for each relevant relationship between an event (marked in yellow) and user role (marked in red) in the ontology

```
@main.route('/listings')
def listings():
    all_listings = Listing.query.all()
    return render_template('listings.html', listings=all_listings)
```

As you modify your web application in Flask, continuously update the ontology to reflect the latest changes. Ensure the database aligns with the evolving design and work Agile!