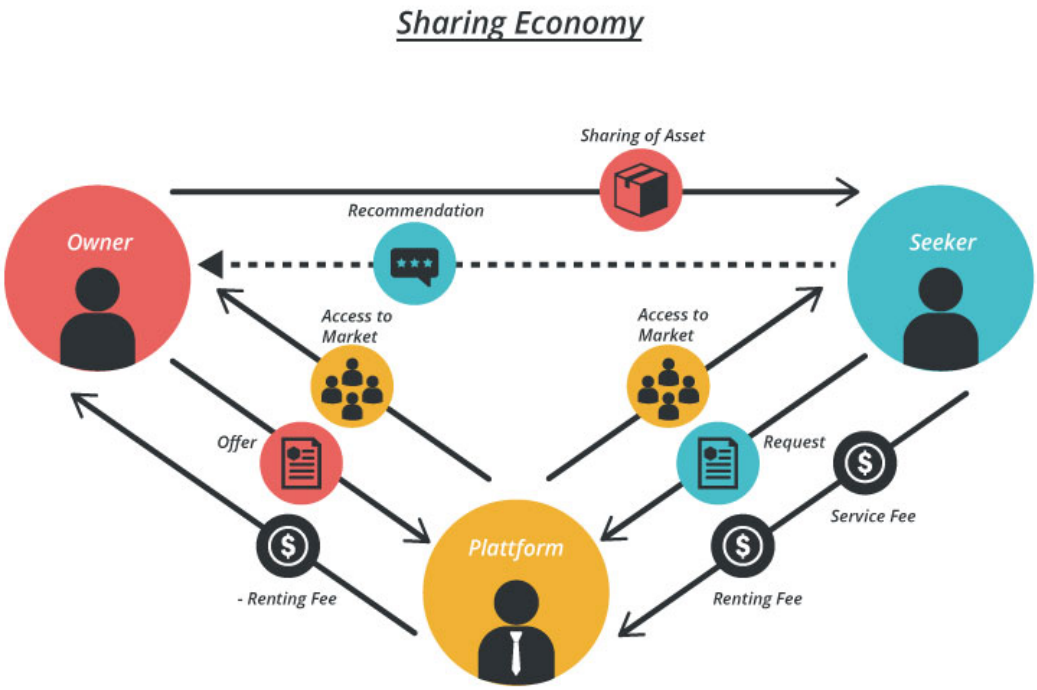
Marketplace Ideation

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



Platform name: KotRepair

Short description:
KotRepair will be an app where students can post when they need help because something is broken at their kot. For example, a problem with the fridge, pump stone, locks, etc. . As a student, you then draw pictures of the problem and describe it as best you can. These posts can then be seen by the right people who can get jobs easier that way. If you think you can solve the problem, you can then click on the post and contact the student. This way the students don't always have to wait for the landlord to solve the problem and there is more work for people.

Model or idea sketch:

We created a logo, we wanted to make it look continuous so we implemented the same colors as in the logo.

Select Properties in the Taxonomy

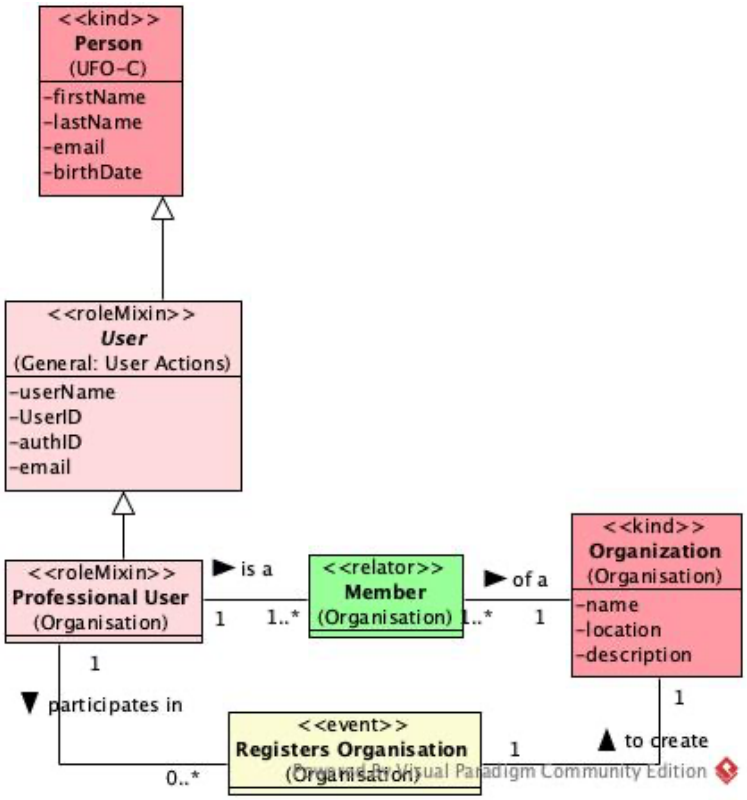
Dimension	Value			
User Type[m]	Person		Organization	
Listing Kind[m]	Good Transfer		Service	
Listing Type[m]	Physical Good[d]	Digital Good[d]	Offline Service[d]	Online Service[d]
Frequency			One-Time[e, d]	Recurring[e, d]
Quantity[m]	One[e]		Many[e]	
Price Discovery	Set by Provider[e]	Set by Customer[e]	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)

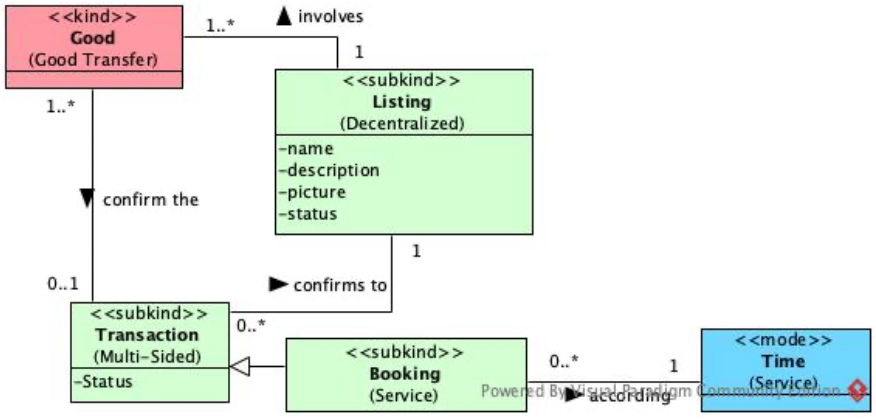
Develop Platform-Specific Ontology

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

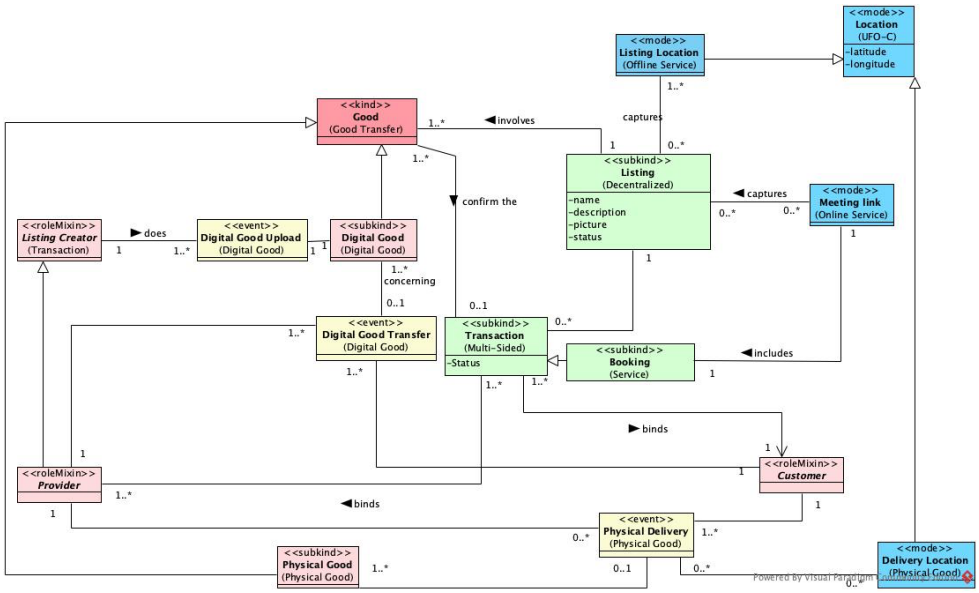
User Type[m]	Person	Organization



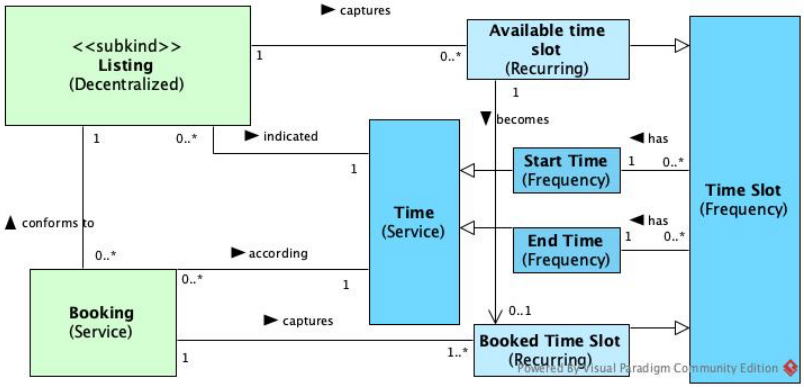
Listing Kind[m]	Good Transfer	Service



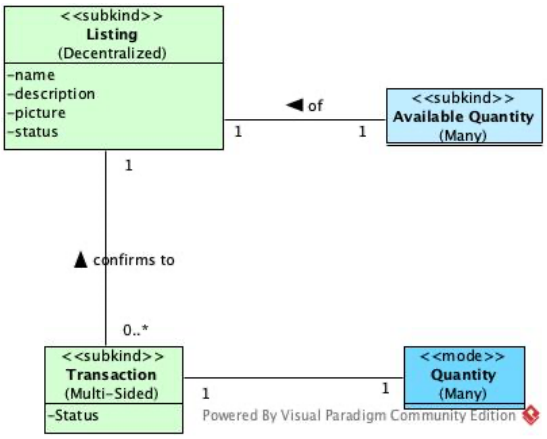
Listing Type[m]	Physical Good[d]	Digital Good[d]	Offline Service[d]	Online Service[d]



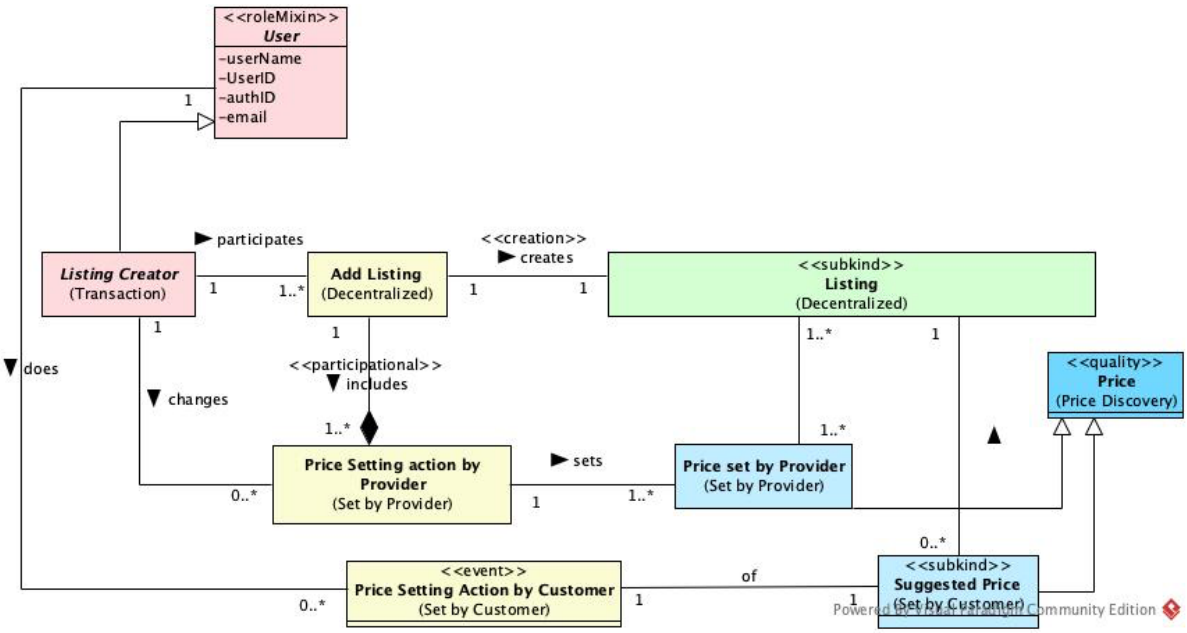
Frequency		One-Time[e, d]	Recurring[e, d]



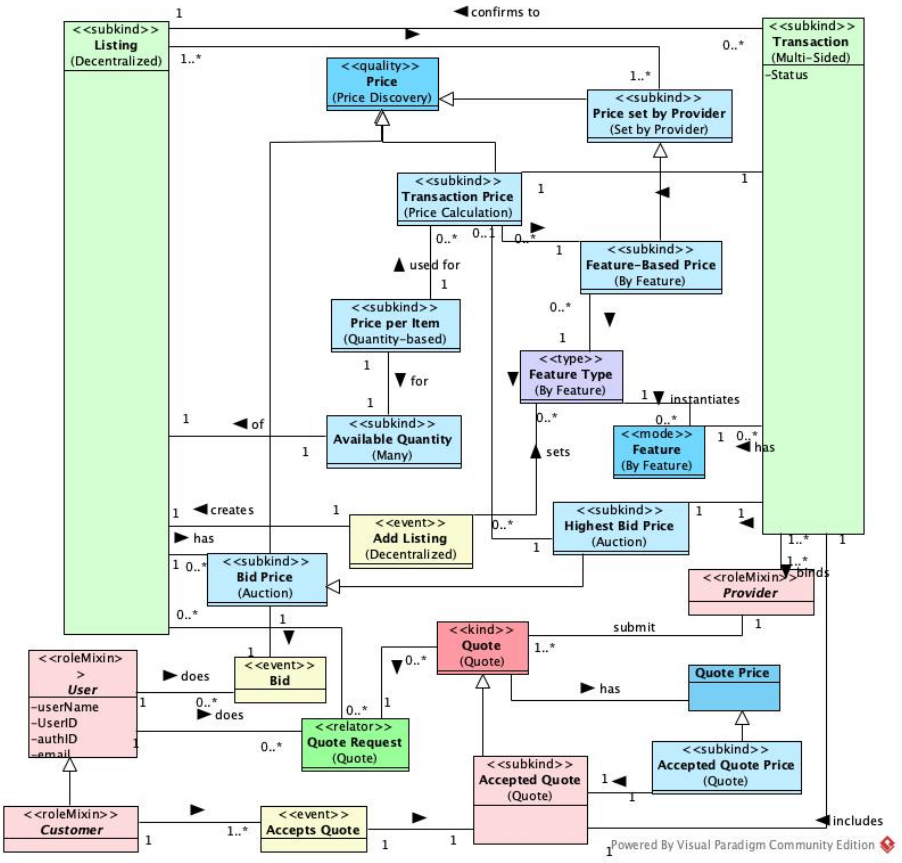
Quantity[m]	One[e]	Many[e]



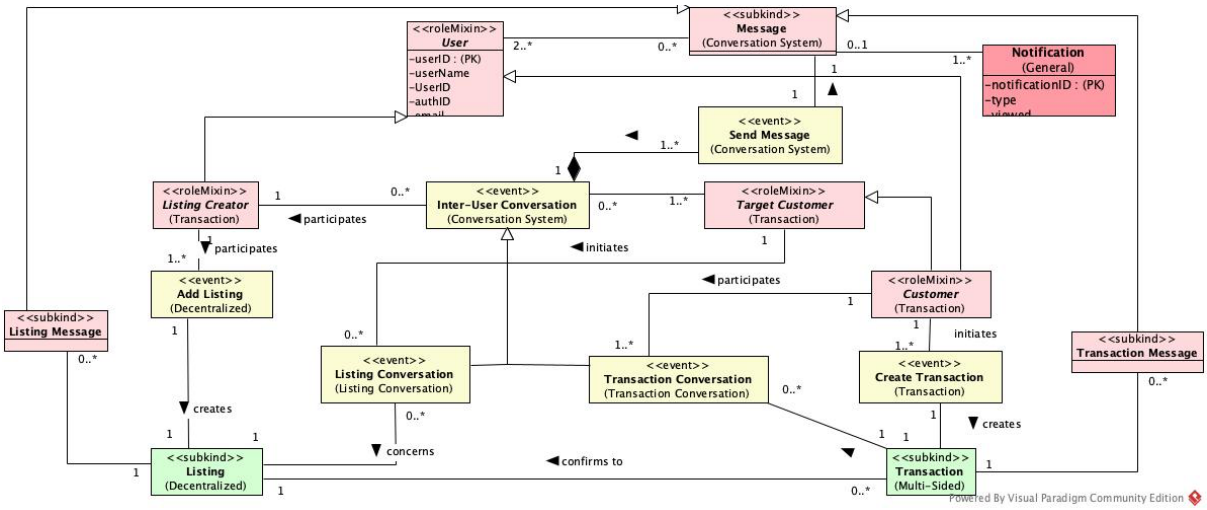
Price Discovery	Set by Provider[e]	Set by Customer[e]	Set by Market[e]



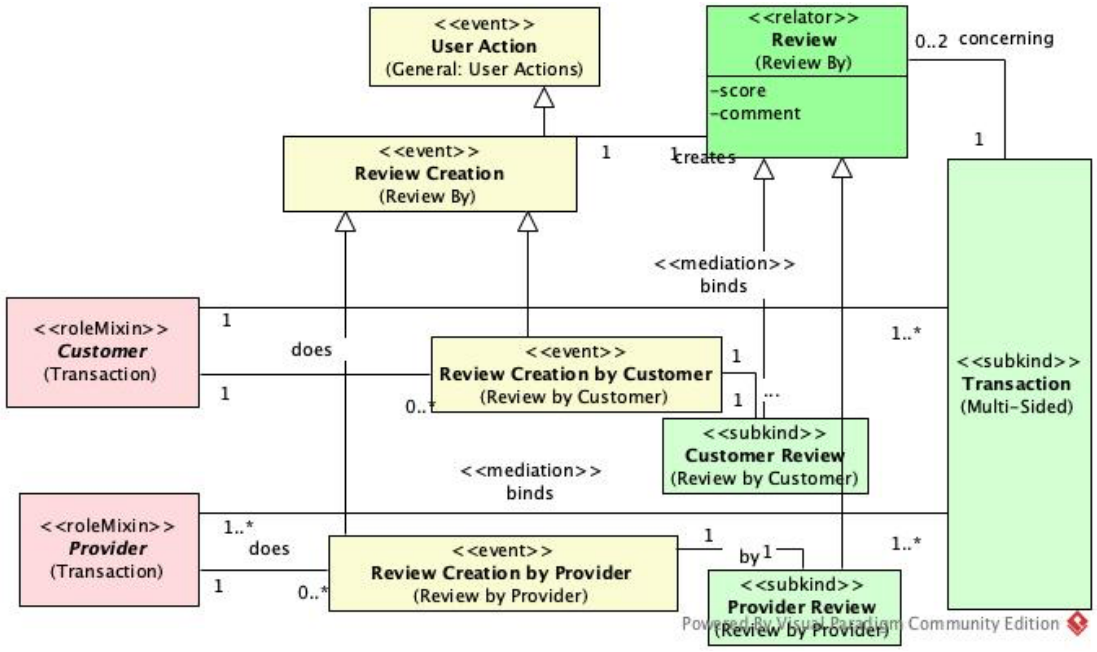
Price Calculation	By Quantity[d]	By Feature[d]	Auction[e, d]	Quote[e, d]



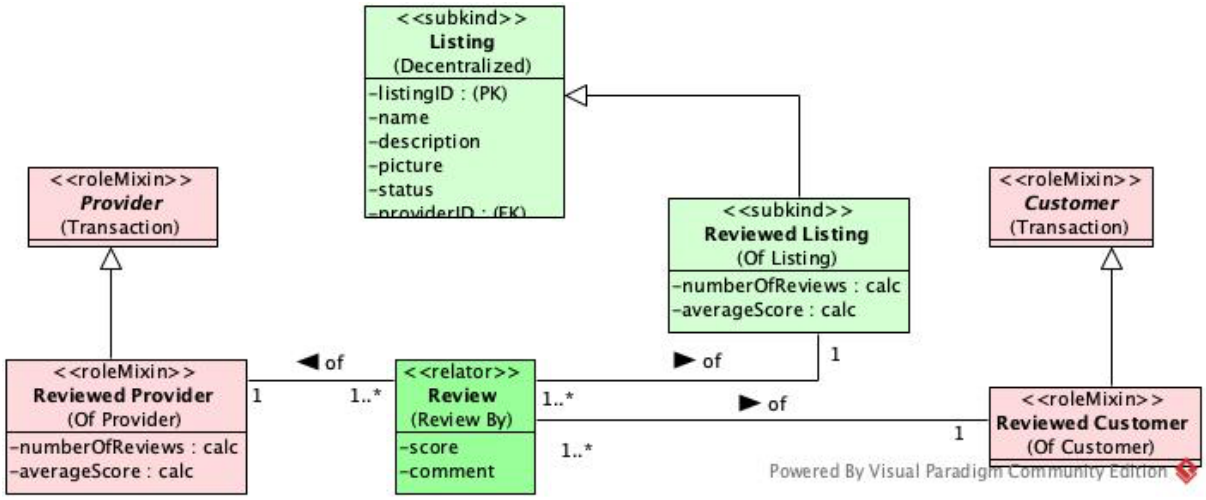
Conversation System	Listing Conversation	Transaction Conversation



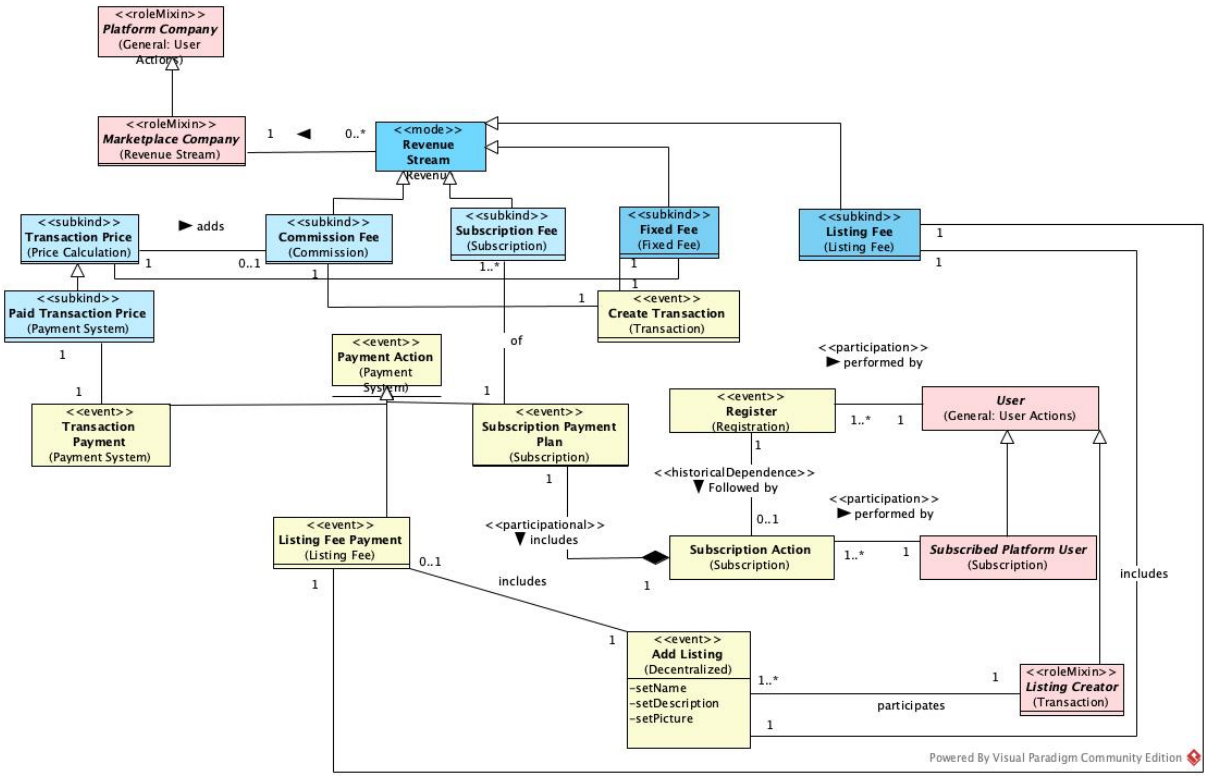
Review by	By Customer	By Provider



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



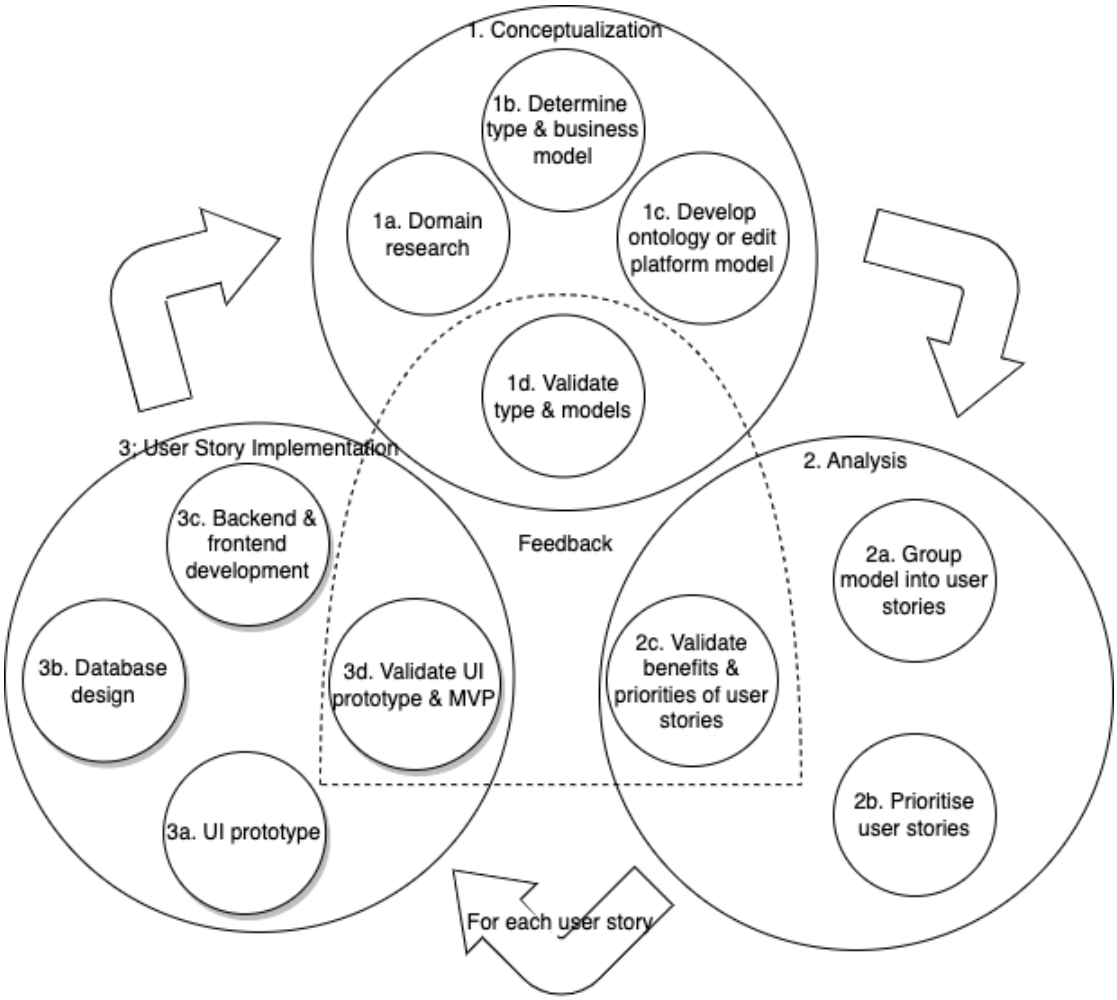
Revenue Stream	Subscription	Commission	Fixed Fee	Listing Fee



Write User Stories

As a (role) *Red class*	**I want to (goal)** *yellow class*	**So that (benefit)**	**Prio**
Student/ technician	Register	I become a user	✓
Technician	Search listings	Find a job	X
Student	Login	I can add a listing and …	X
Student	Adds a listing	his kot can be repared	X
Technician	search a listing	he can earn some money/ repair something	X
Technician	starts a chat	arrange a meeting and discuss price with students	X
Student	remove/ mark the listing as completed	the job is removed from the listing page (completed)	X
Student	pays the money before the repair	the money is on the app	
Student	Gives a rating	Others students can see how good the technician is	X

Ontology-driven MVP development method

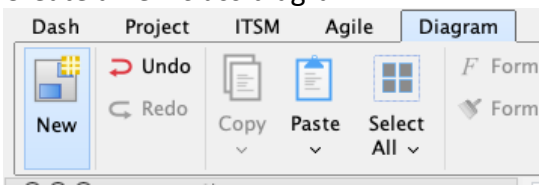


3a: UI Prototype

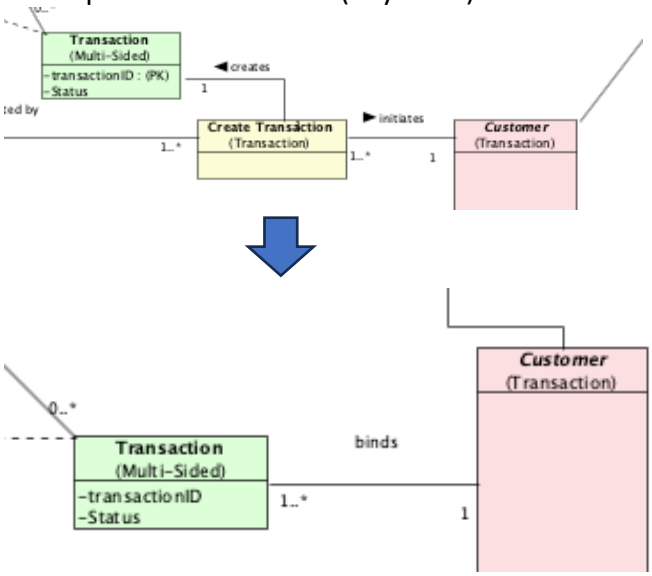
www.figma.com
https://help.figma.com/hc/en-us/categories/360002051613-Get-started
https://www.youtube.com/watch?v=HZuk6Wkx_Eg

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

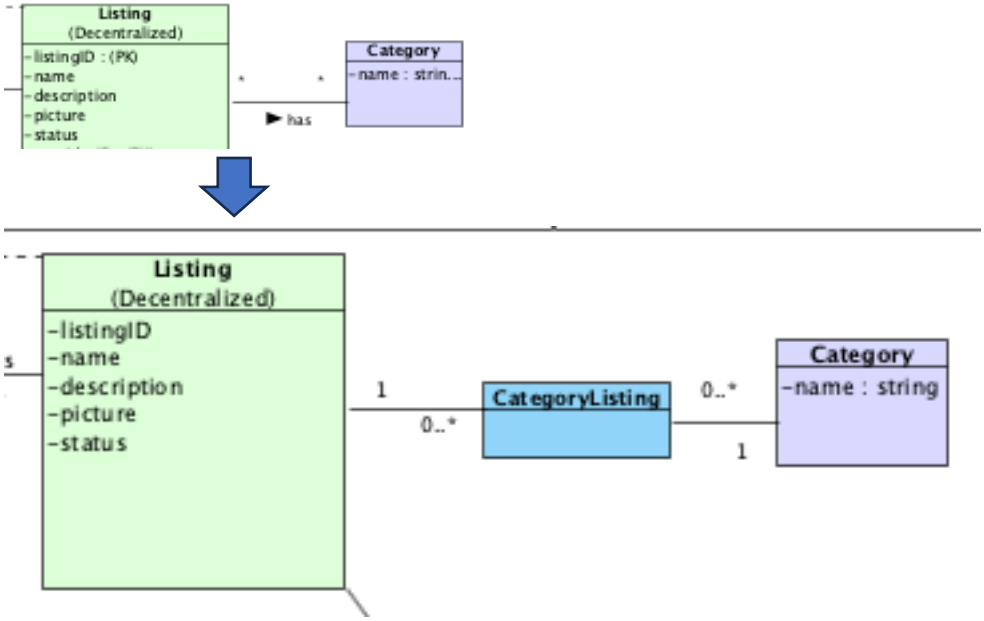
1. Create a new *class* diagram



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

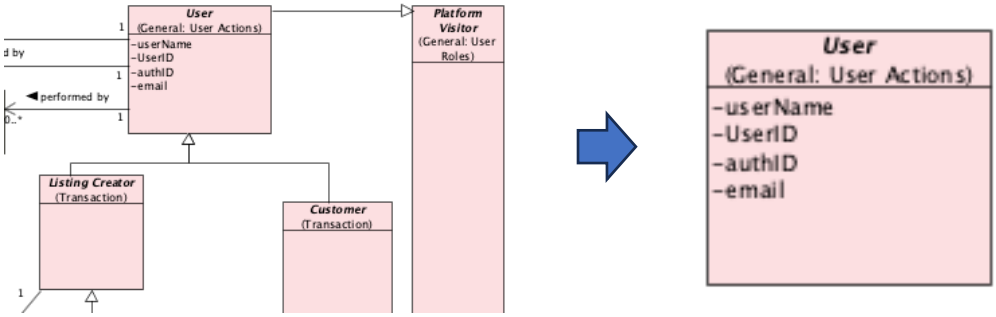


3. Introduce a new table between * to * relationship



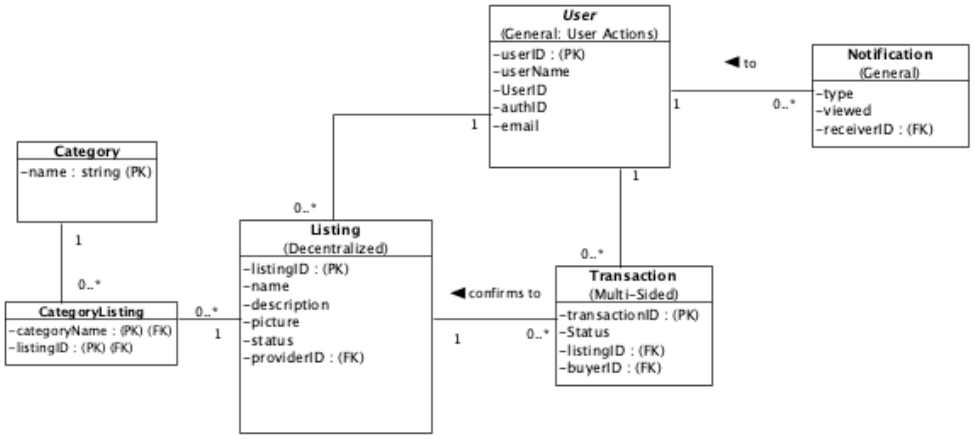
4. Add mode, quality and quantity classes (blue) with a 1 – 1 relationship as attributes

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!