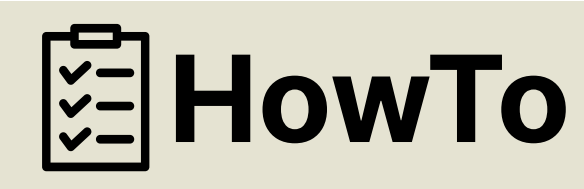
HowTo

How to do anything in the world.



Problem

People often have to wing their way through life, not knowing how to do things. This means that either, they don't do things, or they do them badly which can lead to a lot of frustration and wasted time and resources.

Solution

I propose a web application that will allow those who have done something before to write a structured guide on how to do the said thing, with pictures and videos where necessary. Those who need help with regards to doing the specific thing can then come to the web application and search for the guide on how to do the thing.

In a situation where the said thing is not available, the user can request for a guide on how to do the thing. The request will be sent to the community of users who have done the thing before, and they can then write a guide on how to do the thing.

The web application will also have a rating system where users can rate the guides on how helpful they were. This will help other users to know which guides are the best to use.

And lastly, users will be able to pay the guide executors for their work if they so wish. This will encourage more people to write guides on how to do things.

Some users may not want to follow the guide themselves, so they can hire someone to do the thing for them. This is where the web application will come in. The web application will have a marketplace where users can hire people to do things for them. The people who are hired will then write a guide on how they did the thing, and this guide will be available to other users who want to do the thing. Obviously, the person who was hired will be paid for their work. Users will then review the person who was hired, and this will help other users to know who to hire.

The release of funds for tasks and guides

The money for tasks should be releasable in bulk or individually. Support will be able to filter the tasks that have been completed and then release the funds for those tasks. e.g. mark all the tasks that have been completed and then release the funds for all the tasks that have been completed.

The release of funds will include generation of table of account numbers and amounts that need to be paid out. The support will then be able to download the table and then process the payments from the

bank account of the company that owns the web application. Then support will use that payment schedule when they access the bank account that receives task fees to make the payments, marking the payment as processed, and the users will be able to download the proof of payment. The support will also be able to mark the payments as completed, and the users will be able to see that the payments have been completed. This would've been better if the payments were done automatically, but we will have to do it manually for now because banks are not yet ready to provide APIs for making payments of this nature (bulk and from escrow to the executor).

We will generate a spreadsheet that will follow the standard bank bulk payment format. The spreadsheet will have the following columns: `Account Number`, `Account Holder`, `Amount`, `Reference`, `Bank`, `Branch`, `Branch Code`, `Account Type`.

- The `Reference` will be the `Task ID` or `Guide ID` that the payment is for.
- The `Bank` will be the bank name of the account holder.
- The `Branch` will be the branch name of the account holder.
- The `Branch Code` will be the branch code of the account holder.
- The `Account Type` will be the type of account that the account holder has.
- The `Account Holder` will be the name of the account holder.
- The `Account Number` will be the account number of the account holder.
- The `Amount` will be the amount that needs to be paid to the account holder.
- The `Task ID` or `Guide ID` will be the unique identifier of the task or guide that the payment is for.
- The `Bank` will be the bank name of the account holder.
- The `Branch` will be the branch name of the account holder.
- The `Branch Code` will be the branch code of the account holder.
- The `Account Type` will be the type of account that the account holder has.

Functional Requirements

1. Users should be able to create an account on the web application.
2. Users should be able to write a guide on how to do something.
   2.1. The guide should be structured in a way that is easy to follow.
   2.2. The guide should have pictures and videos where necessary.
   2.3. The guide should be tagged with the thing that it is about.
   2.4. The guide should have steps that represent an action that the user should take at each point in the guide.
3. Users should be able to search for a guide on how to do something.
4. Users should be able to request a guide on how to do something.
5. Users should be able to rate a guide on how helpful it was.
6. Users should be able to pay the guide executors for their work.
7. Users should be able to hire someone to do something for them.
8. Users should be able to write a guide on how they did something for someone else.
9. Users should be able to review the person who was hired to do something for them.
10. The web application should have a marketplace where users can hire people to do things for them.

Non-Functional Requirements

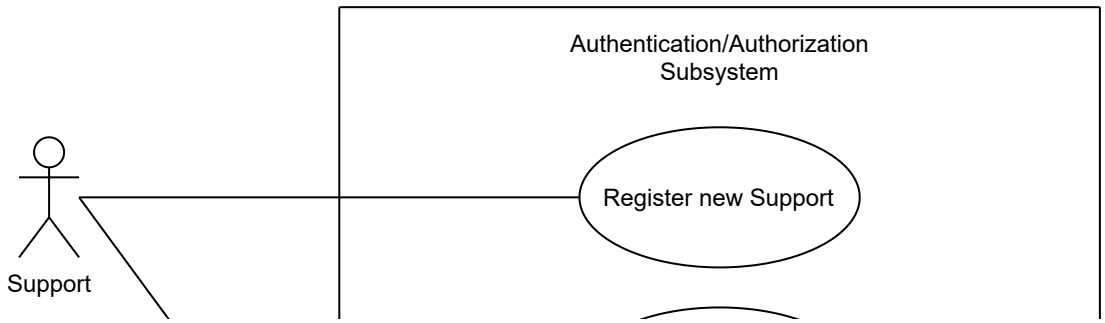
1. The web application should be easy to use, and this will be measured by the number of users who are able to write a guide on how to do something.
2. The web application should be fast, and this will be measured by the time it takes for a user to search for a guide on how to do something.
3. The web application should be secure, and this will be measured by the number of security breaches that occur.
4. The web application should be reliable, and this will be measured by the number of times the web application goes down.
5. The web application should be scalable, and this will be measured by the number of users who are able to use the web application at the same time.

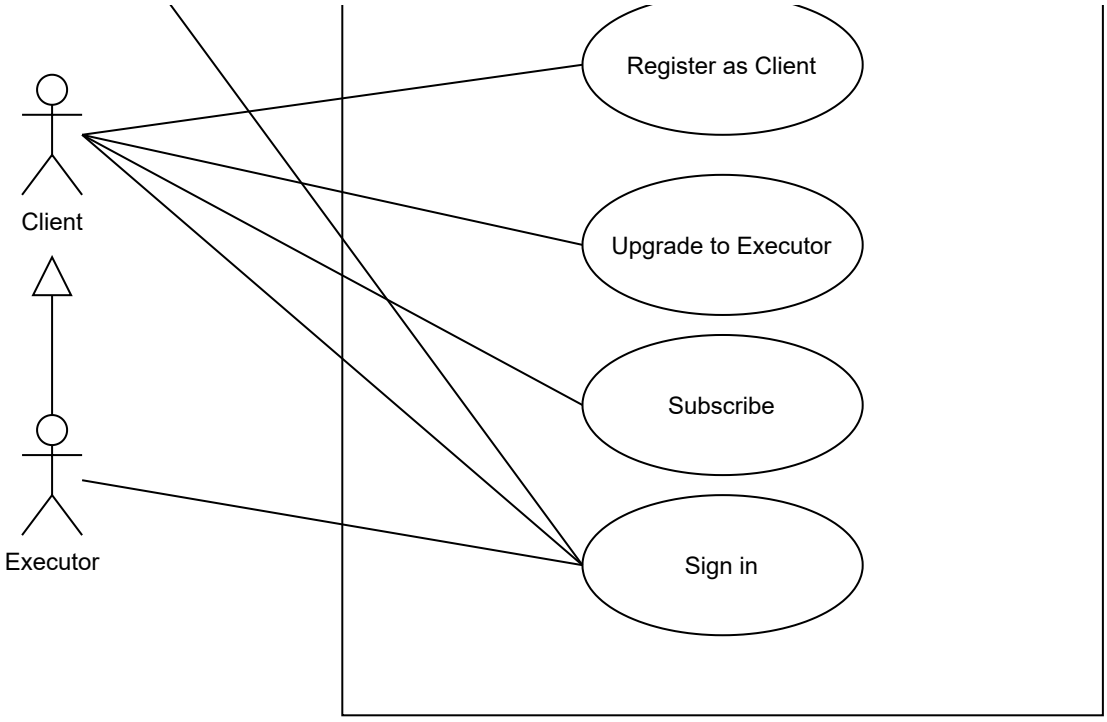
Users

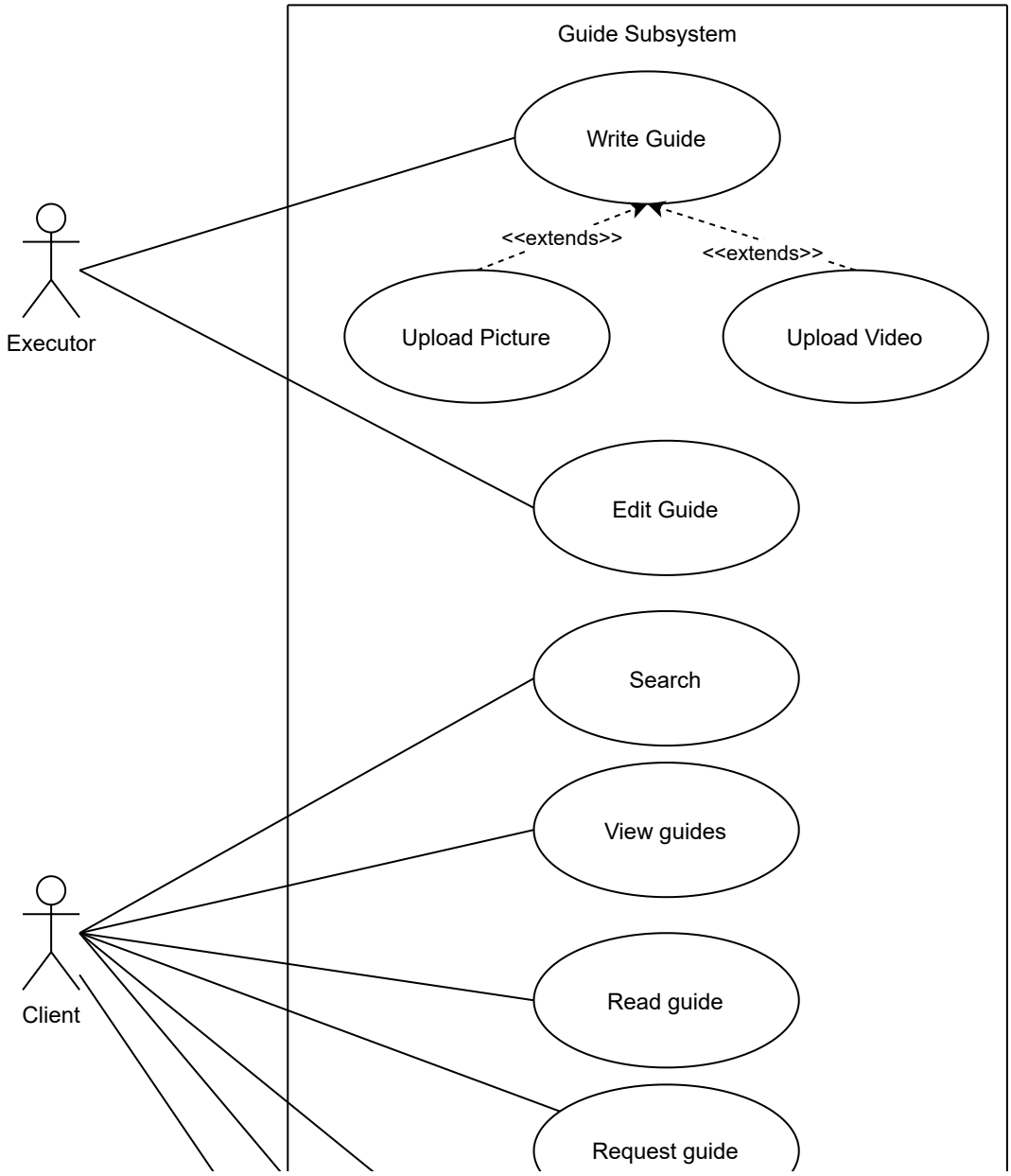
1. Guide Executors: Users who have done something before and want to write a guide on how to do the thing.
2. Guide Clients: Users who need help with doing something and want to read a guide on how to do the thing.
3. Guide Requesters: Users who need help with doing something and want to request a guide on how to do the thing.
4. Guide Raters: Users who want to rate a guide on how helpful it was.
5. Guide Payers: Users who want to pay the guide executors for their work.
6. Thing Doers: Users who want to hire someone to do something for them.
7. Thing Doer Executors: Users who have been hired to do something and want to write a guide on how they did the thing.
8. Thing Doer Reviewers: Users who want to review the person who was hired to do something for them.

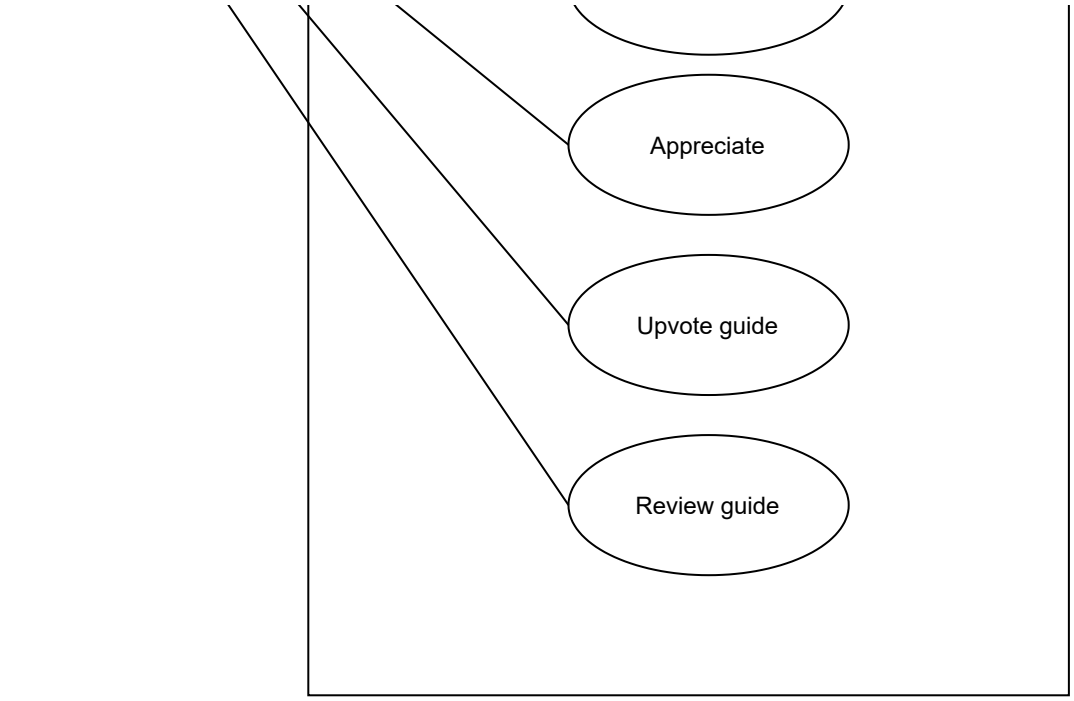
The web application should have a clear distinction between the different roles that a user can have. So that a user can easily switch between the different roles that they have. The main dinstinct user types are Executor[E],Client[C], and the Support[S] users. The assumption that we will make is that the Executor can write the guide and execute tasks, but they can also act as a Client, The Client cannot write reviews, or execute tasks. Client can only read the guides and execute their own tasks. The Executor user can do all that a Client can do and also write reviews, execute tasks, and write guides. The Support user can do all that can be done in the system, but they are only people who work for the company that owns the web application. Clients can upgrade to Executors based on the amount of guides that they have read, they can be put on probation by allowing them to write guides that will be reviewed by the Support users. The Support users can also downgrade the Executors to Clients if they are not writing good guides.

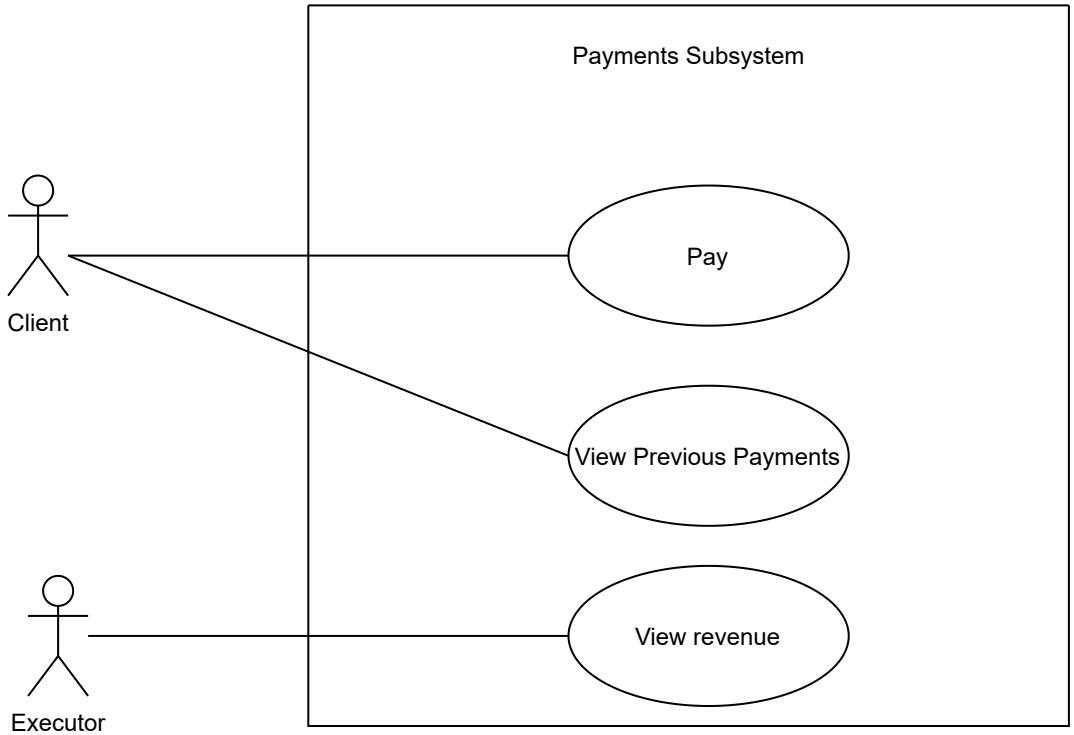
Use Cases

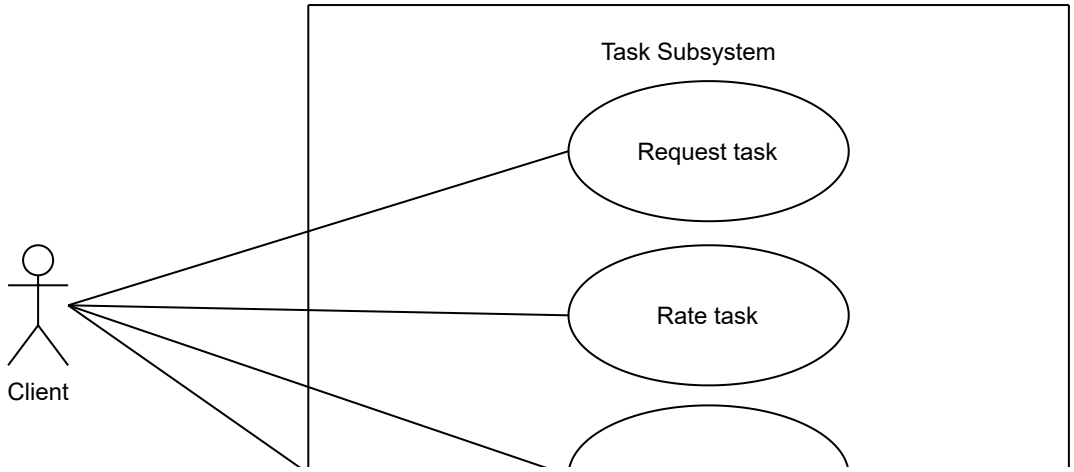


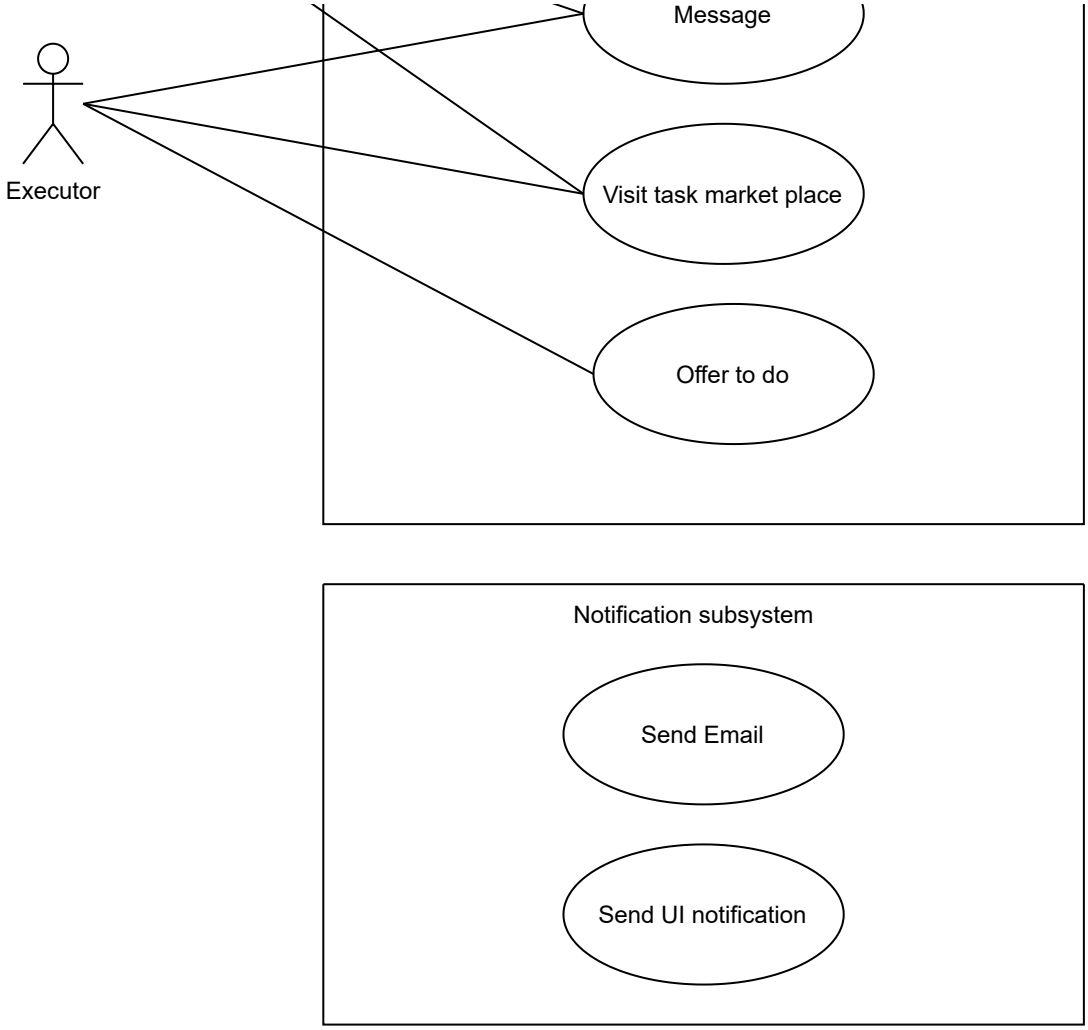

Client
Executor
Register as Client
Upgrade to Executor
Subscribe
Sign in


Guide Subsystem
Executor
Write Guide
<<extends>>
<<extends>>
Upload Picture
Upload Video
Edit Guide
Search
View guides
Client
Read guide
Request guide

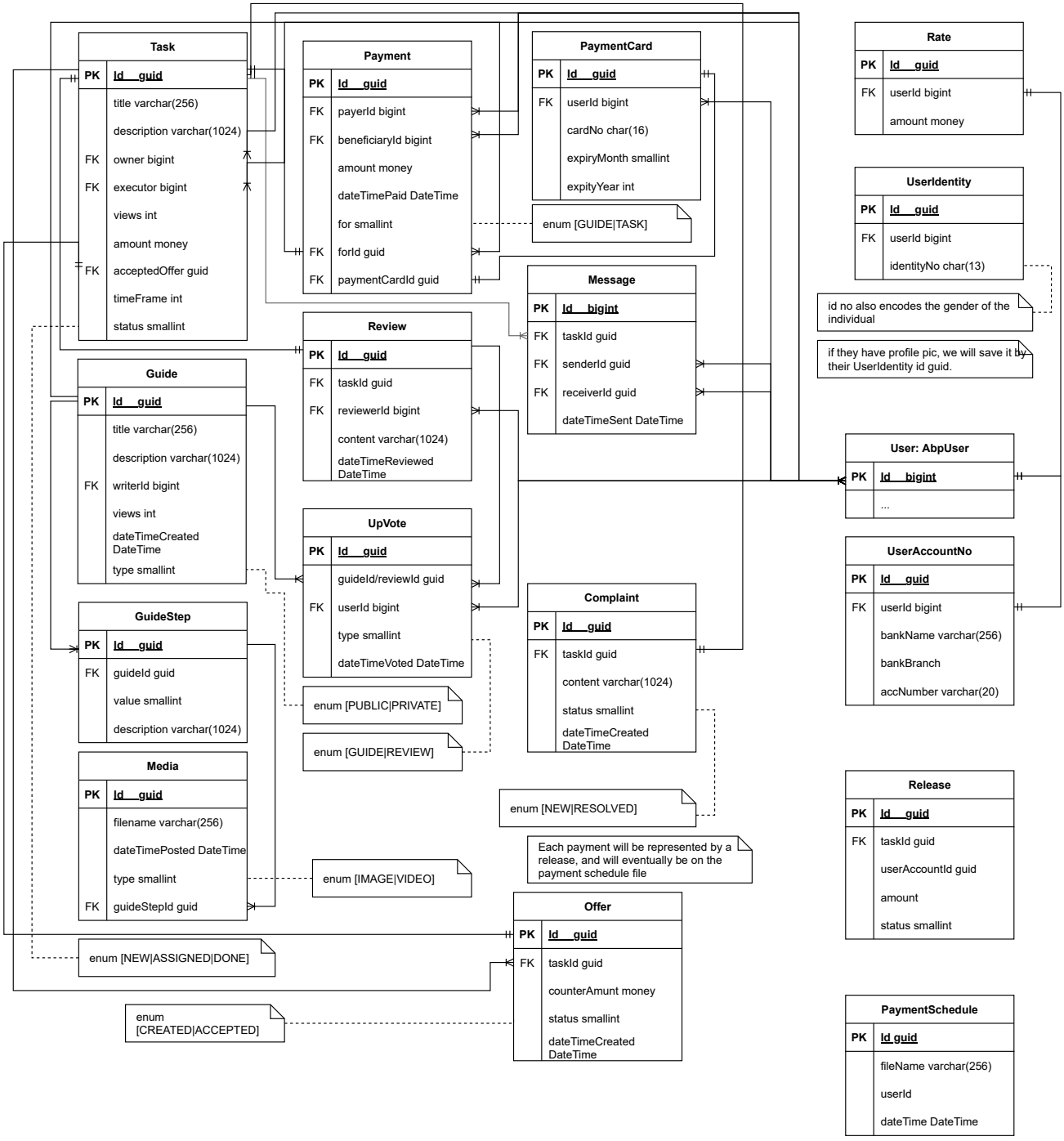

Appreciate
Upvote guide
Review guide


Payments Subsystem
Client
Pay
View Previous Payments
Executor
View revenue


Task Subsystem
Request task
Client
Rate task

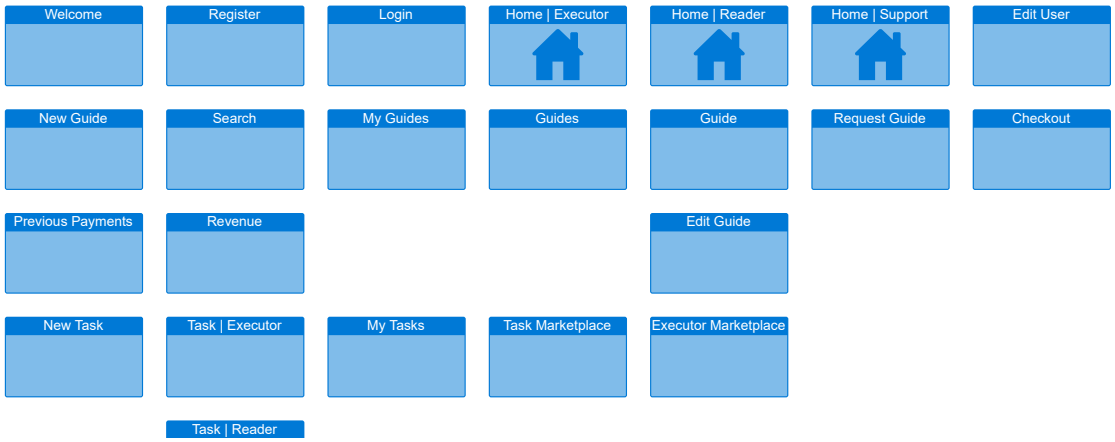


Domain Model

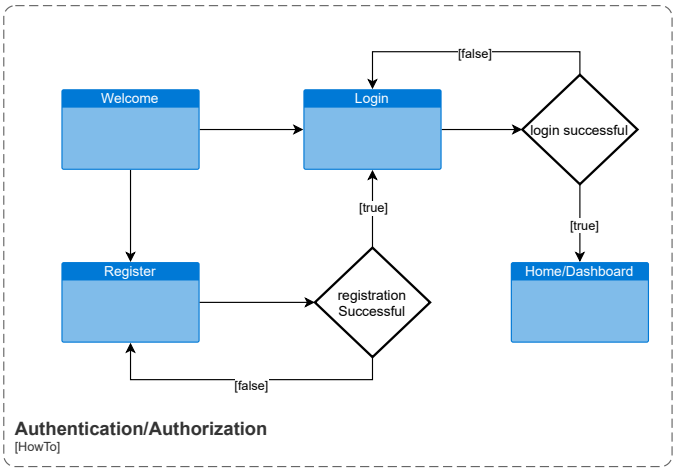


User Flows

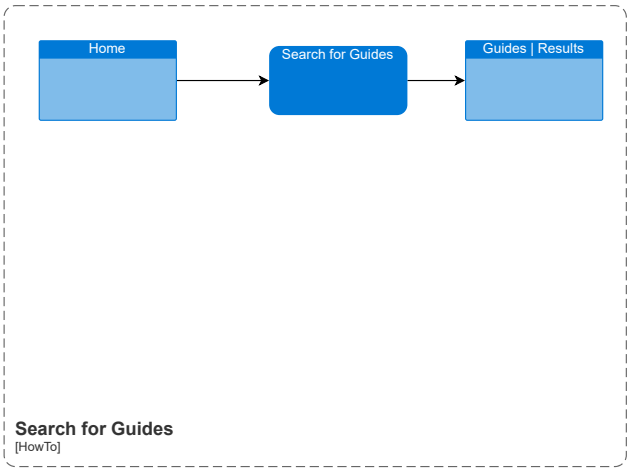
PAGES



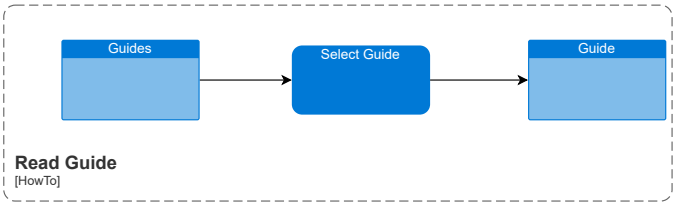
USER FLOWS



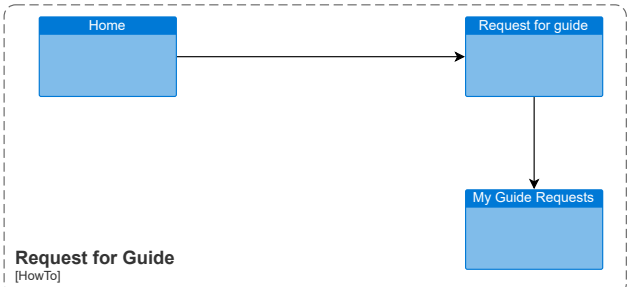
Authentication/Authorization
[HowTo]



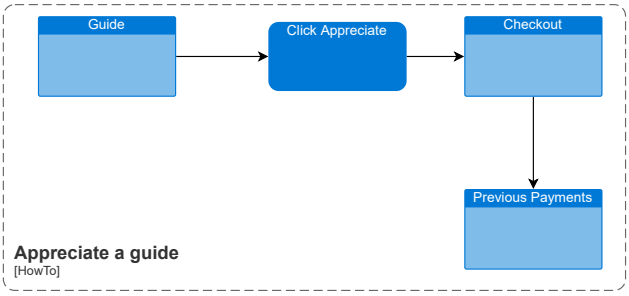
Search for Guides
[HowTo]



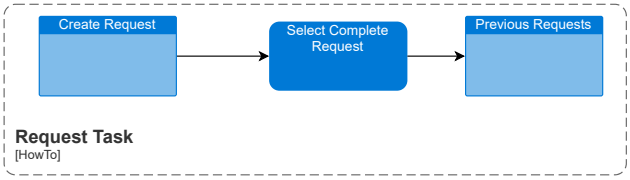
Read Guide
[HowTo]



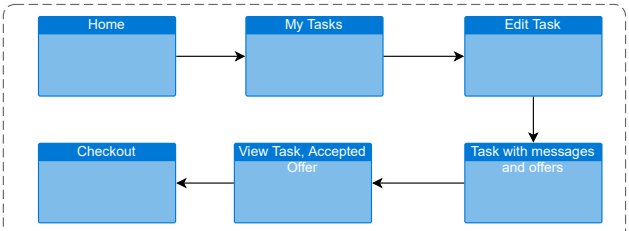
Request for Guide
[HowTo]



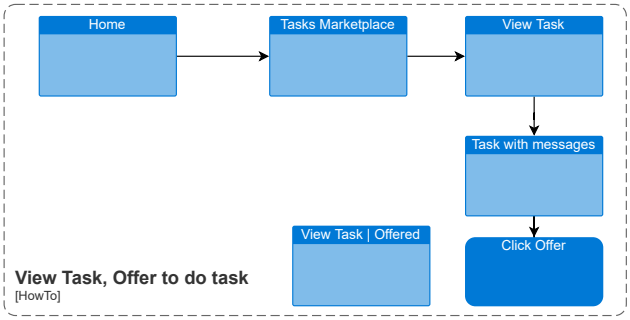
Appreciate a guide
[HowTo]



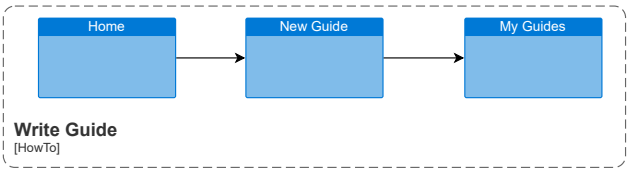
Request Task
[HowTo]



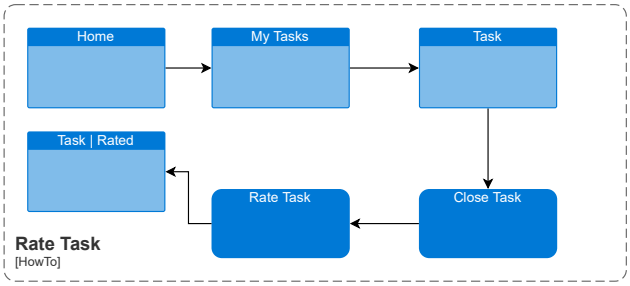
Edit Task, Accept offer
[HowTo]



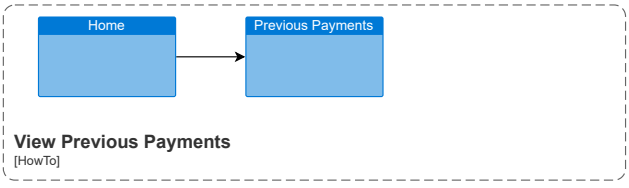
View Task, Offer to do task
[HowTo]



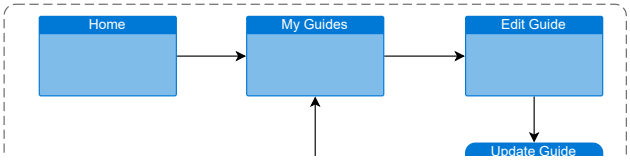
Write Guide
[HowTo]

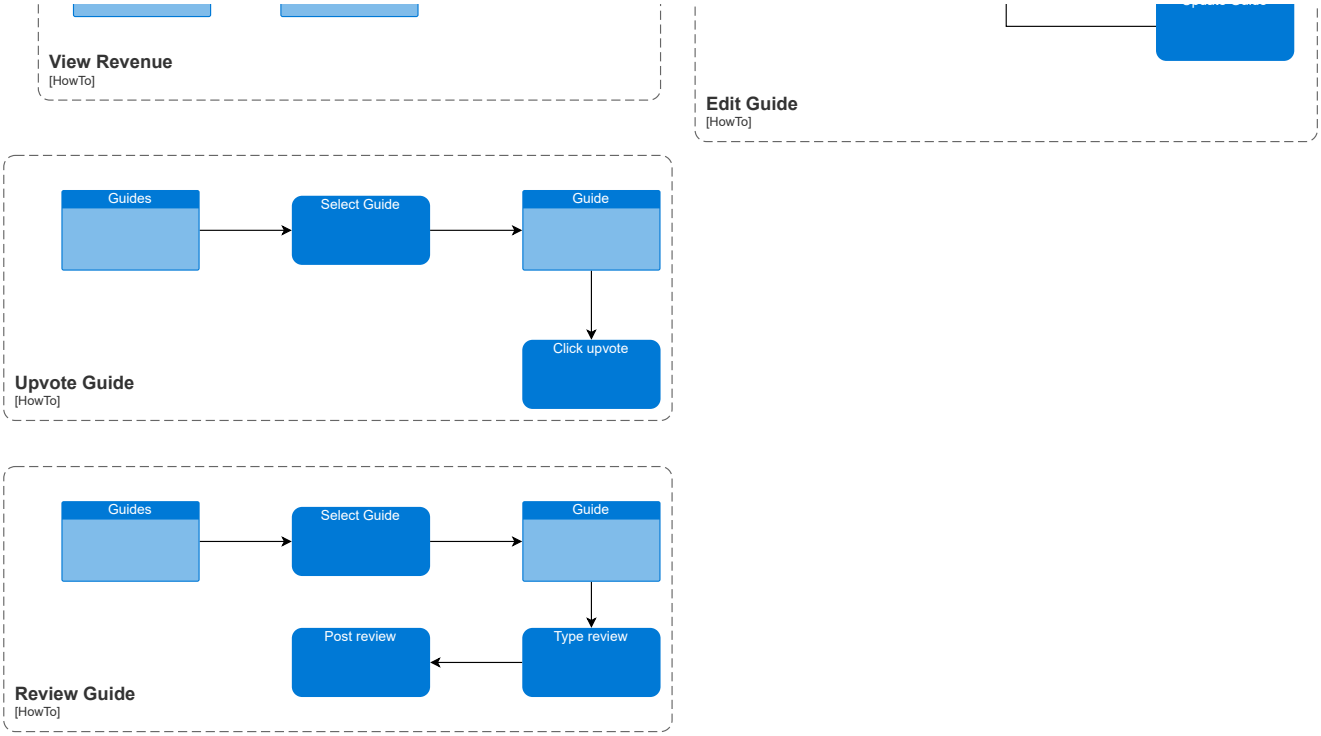


Rate Task
[HowTo]



View Previous Payments
[HowTo]





Technology Stack

1. Frontend: Next.js
2. Backend: ABP Framework
3. Database: Microsoft SQL Server
4. Hosting: Azure App Service (Backend) + Vercel (Frontend)
5. Payment Gateway: Payfast
6. Video Hosting: Azure Blob Storage
7. Image Hosting: Azure Blob Storage
8. Search: Azure Cognitive Search ??
9. Email: Azure Communication Services
10. Authentication: Azure Active Directory ??
11. Version Control: GitHub