System Specification

Project Name	Ember Exchange
Project Manager	Franz Auernig ?
Document Owner	Arbeitszeitbetrug
Created on	19/12/25
Last modified	15/01/26
State	In Planning
Dokumentablage	SystemSpecificationTemplate.doc

Document History

Revision			Chapter	Modification	Author
Nr.	Date	Version			
1	15.01.2026	0.1	All	Initial Draft	Arbeitszeitbetrug

Content

1 Initial Situation and Goal

1.1 Initial Situation

- Pain of status quo

What sucks about today's online games?

Pay-to-win, random price dynamics, slow progress, etc.

- Project Trigger

After the Clash Royale Hero Update, we realized how few good online games that provide fun without having to spend money there are.

- Expected Win

Players gain fun and everybody can access simulated economy data

1.1.1 Application Domain

Core Idea (layman)
EmberExchange is a “virtual flea-market for stove pictures”. People open digital surprise boxes, get stove images, and trade them like collectible cards—while the computer writes down every price so everyone can see how rarity changes value.

Where it lives
- **Physical layer**: any laptop with a browser and Node.js.
- **Organizational layer**: Twitch stream, or hobby server—no money, no company, no legal gambling license needed.

Domain terms
a. **Lootbox**: digital sealed envelope that drops a random stove.
b. **Stove rarity**: common / rare / mythic / legendary / limited tag that decides drop chance.
c. **Ownership chain**: list of former owners, like a car title.
d. **Price history**: line graph of past sale prices for one stove type.

Business process
1. Player registers → 2. Receives starter coins → 3. Opens lootboxes → 4. Lists items for sale → 4b. Optional faucet: player starts a mini-game (match-3, roulette, etc.) and wins a small fixed coin payout; → 5. Others bid/buy → 6. Ownership & price are logged → 7. Player can reset the whole world for a new experiment.

Hard facts vs. Assumptions
Facts: SQLite file stores every trade; no real money;
Assumptions: players trust public ledger;

1.1.2 Glossary

Lootbox
A free, limited-quantity container that opens to reveal one random stove. Drop probabilities are public and no real money is required.

Stove
A purely cosmetic digital collectible represented by a 64×64 PNG and an ID; it has no gameplay stats.

Rarity Tier
One of common, rare, mythic, legendary, limited; stored in the stove record and used to set lootbox drop rates.

Ownership Chain
An append-only list kept in the database that records every previous owner of a given stove.

Price History
A time-series table that stores the coin price and timestamp of each completed sale for a stove type.

Coin
The in-game currency; earned through starter grant, mini-games, or selling stoves; has no cash value.

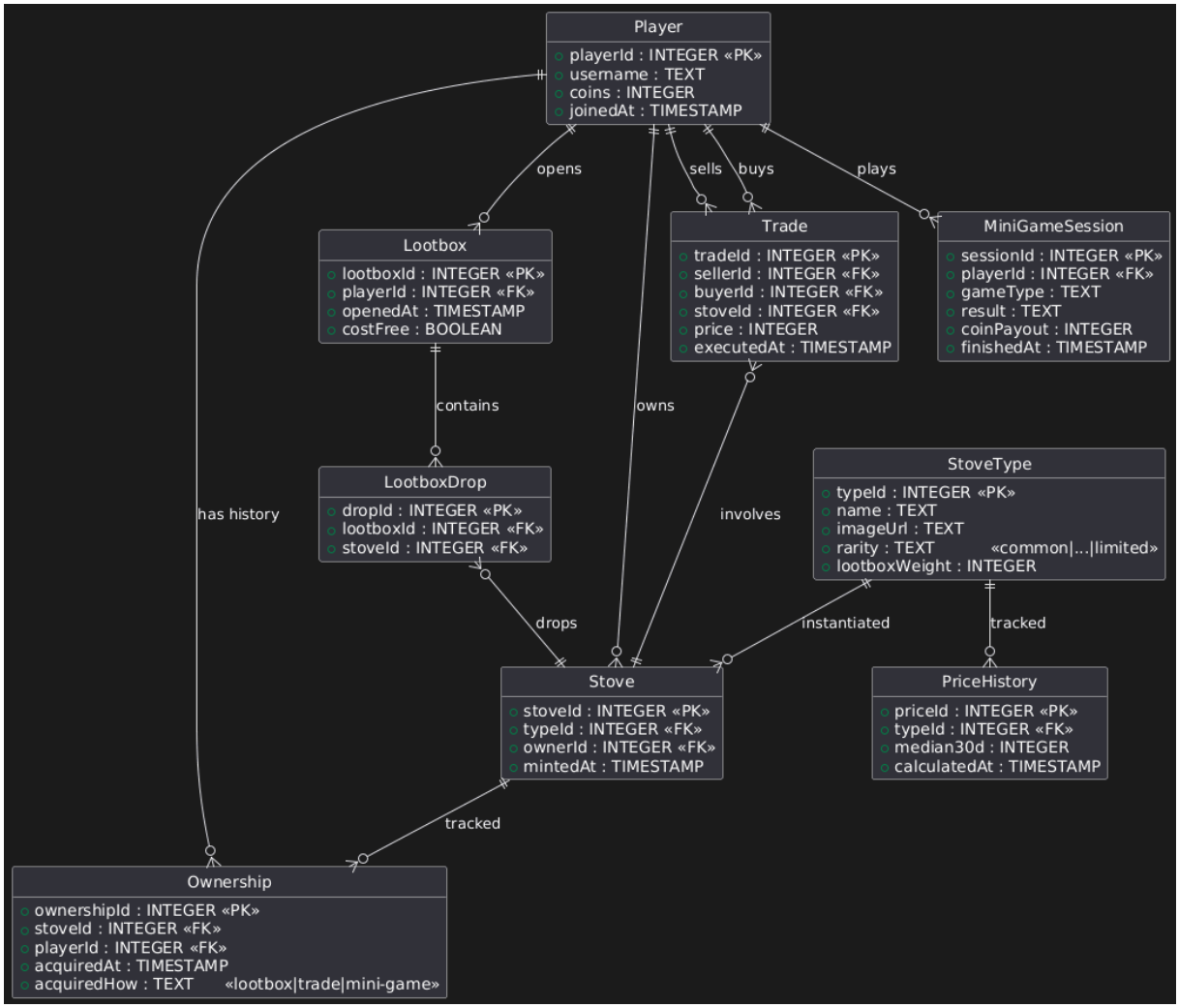
Mini-Game Faucet
Optional skill or chance game (match-3, roulette, etc.) that pays out small, fixed coin rewards

Marketplace
The web page where players place buy/sell orders and where median prices are calculated automatically.

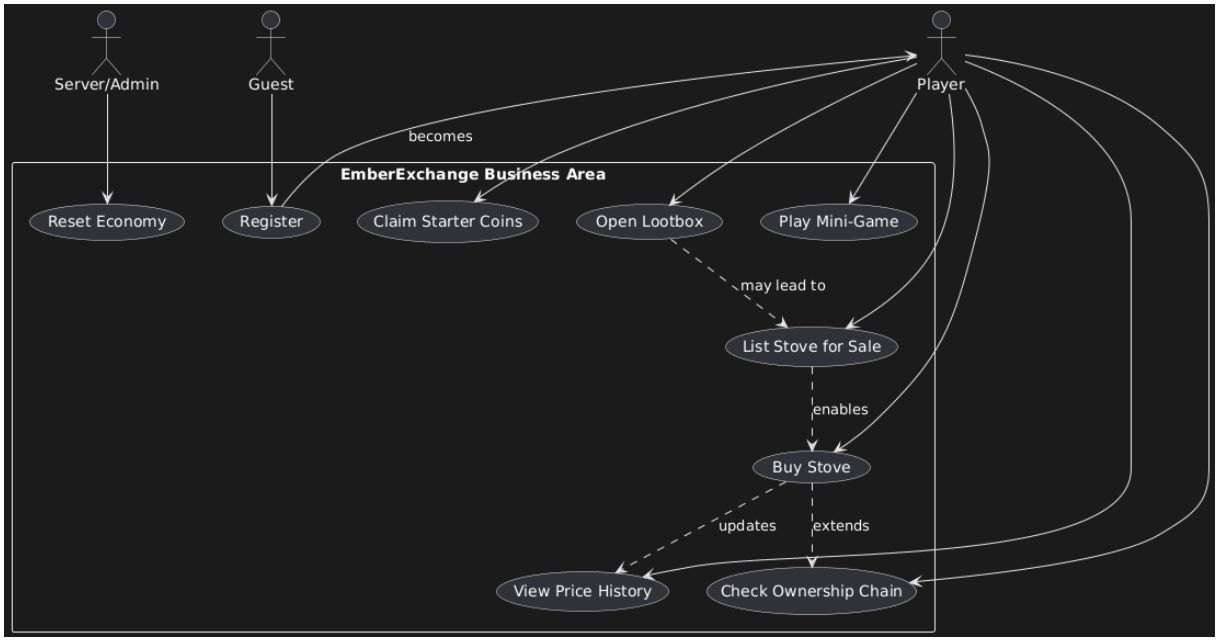
Median Pricing
The system uses the last 30-day median sale price to suggest fair value and to limit obvious price manipulation.

Reset Function
A one-click admin command that wipes all inventories, trades, and price data so the economy can be rerun.

1.1.3 Model of the Application Domain



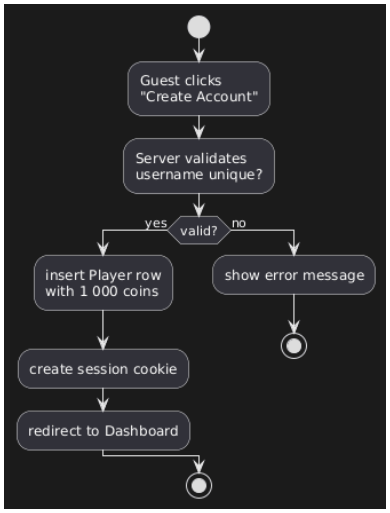
1.1.4 Overview of the Business Processes



1.1.5 Description of the Business Processes

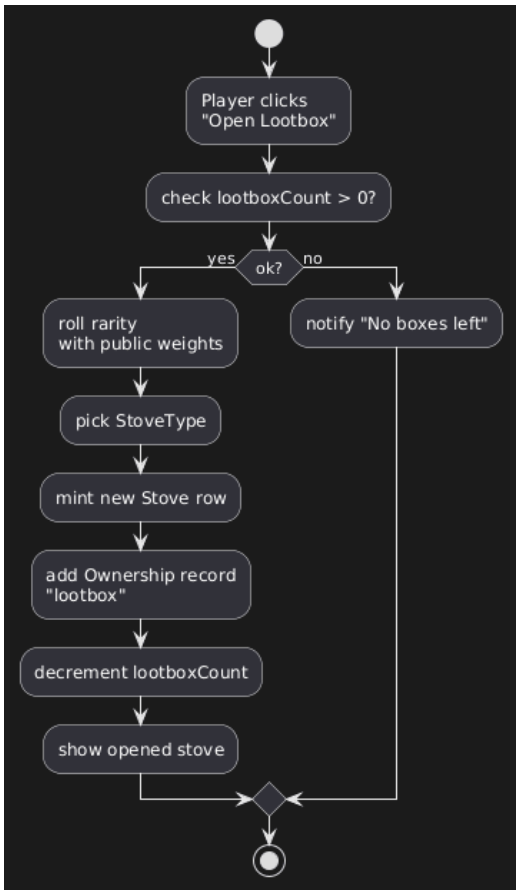
Description of BP-1: Player Registration & On-Boarding

Triggering Event:	Guest clicks “Create Account” and submits a username.
Result:	New player records with 1 000 starter coins; user is logged in and can use all features.
Contributors:	Guest, EmberExchange web server, SQLite player table.



Description of BP-2: Lootbox Opening

Triggering Event:	Player presses “Open Lootbox” while an unused box is available.
Result:	One random stove created & assigned to player; ownership log updated; box counter decremented.
Contributors:	Player, Lootbox service, Stove-type table, Ownership ledger, SQLite.



1.2 Goal Definition

Main Goal

Give students, streamers and hobbyists a **zero-cost, transparent sandbox** that lets them *experience* supply-and-demand price formation and *inspect* every transaction, so they can learn how virtual economies work without being exposed to pay-to-win or real-money gambling.

Reason for Development

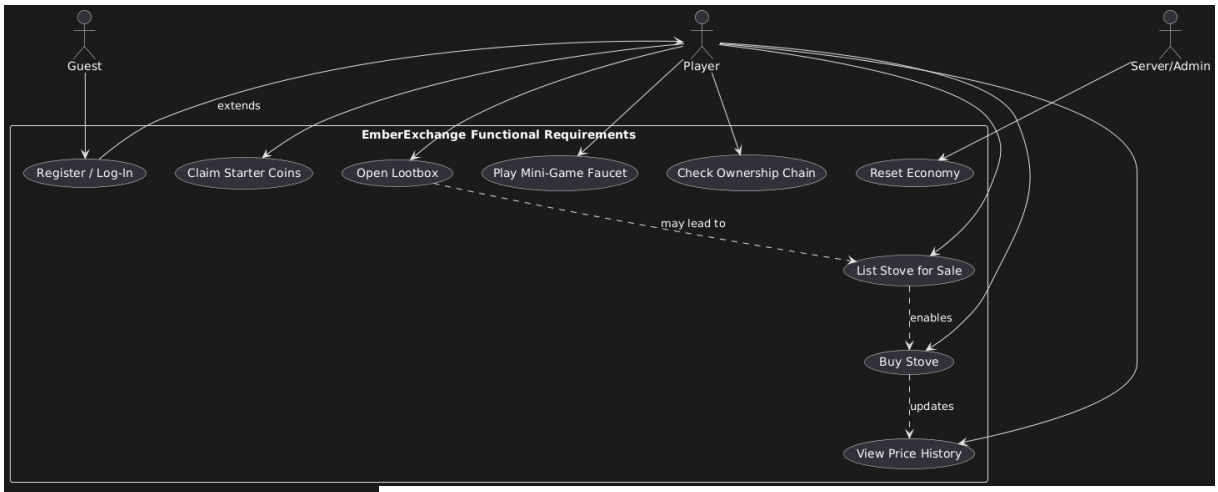
Existing games hide their market data; there is no lightweight, resettable demonstration kit that reveals rarity tables, ownership chains and median price curves in real time. EmberExchange fills that gap and can be spun up on a laptop in under five minutes.

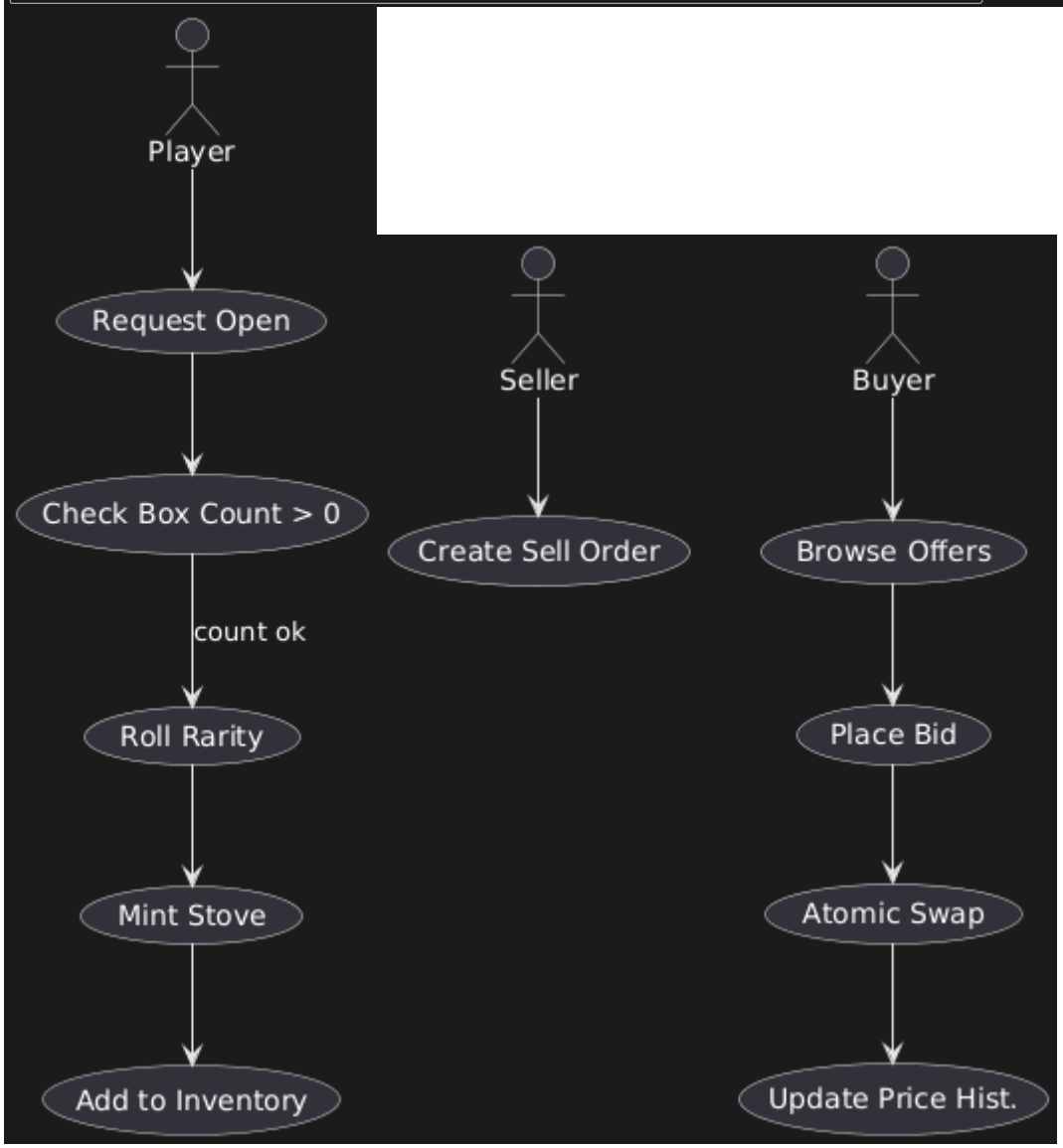
Target Group & Prerequisites

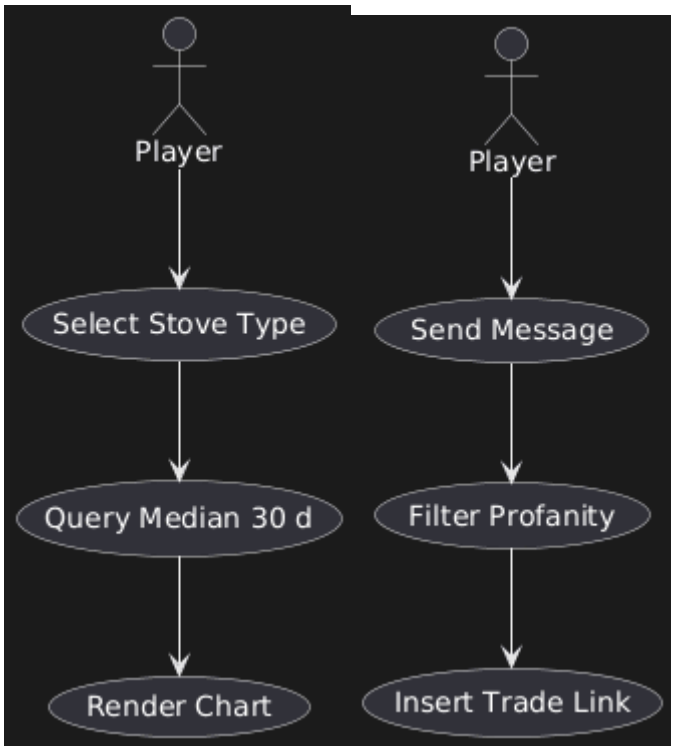
- **Primary**: computer-science students and lecturers (basic SQL/JS knowledge, can read a REST endpoint).
- **Secondary**: Twitch/YouTube creators who want a plug-in collectible mini-game (no coding, just run).
- **Assumed**: familiarity with browser UI, copy-paste terminal commands, and the idea of “lootboxes” from mainstream games; no finance or blockchain background required.

2 Functional Requirements

2.1 Use Case Diagrams


Guest
Player
Server/Admin
extends
EmberExchange Functional Requirements
Register / Log-In
Claim Starter Coins
Open Lootbox
Play Mini-Game Faucet
Check Ownership Chain
Reset Economy
may lead to
List Stove for Sale
enables
Buy Stove
updates
View Price History


Player
Request Open
Check Box Count > 0
count ok
Roll Rarity
Mint Stove
Add to Inventory
Seller
Create Sell Order
Buyer
Browse Offers
Place Bid
Atomic Swap
Update Price Hist.



2.2 Use Cases

Use-Case ID: UC-1 | Register / Claim Starter Coins
1. Brief: Guest picks unique username; system creates account with 1 000 coins and logs in.
2. Actors: Guest → Player
3. Pre: none
4. Post: Player row, balance 1 000, session active
5. Main: submit → unique check → create → cookie → dashboard
6. Alt: name taken → retry
7. Exc: network → retry msg
8. Freq: any time
9. Spec: <1 s, 20 char, alnum+_

Use-Case ID: UC-2 | Open Lootbox
1. Brief: Consume one lootbox, receive random stove.
2. Actors: Player
3. Pre: lootboxCount > 0
4. Post: new Stove row, ownership log, lootboxCount -= 1
5. Main: click → roll rarity → insert stove → update inventory
6. Alt: zero boxes → message
7. Exc: concurrent → serial tx
8. Freq: often
9. Spec: <1 s, drop odds public

Use-Case ID: UC-3 | List & Buy Stove
1. Brief: Seller lists, buyer purchases; atomic swap.
2. Actors: Seller, Buyer
3. Pre: Seller owns stove; Buyer has enough coins

4. Post: ownership & coins swapped; price history updated
5. Main: list → browse → buy → atomic tx → done
6. Alt: insufficient coins → error
7. Exc: double click → first wins
8. Freq: often
9. Spec: <2 s, median recalc

Use-Case ID: UC-4 | View Price History
1. Brief: Show 30-day median price chart for a stove type.
2. Actors: Player
3. Pre: none
4. Post: chart rendered
5. Main: select type → query median → display
6. Alt: no trades → empty chart
7. Exc: none
8. Freq: any time
9. Spec: <1 s

Use-Case ID: UC-5 | Check Ownership Chain
1. Brief: List previous owners of a stove.
2. Actors: Player
3. Pre: valid stoveId
4. Post: timeline displayed
5. Main: enter id → fetch chain → show
6. Alt: not found → error
7. Exc: none
8. Freq: occasional
9. Spec: <1 s

2.2.1 Characteristic Information

Superior business process:	BP-1 Player Registration & On-Boarding
Goal:	Create a playable account with starting capital so trading can begin immediately.
Precondition:	Guest has network access; server online; username not yet taken.
Postcondition:	Player row exists with 1 000 coins; valid session cookie; redirected to dashboard.

Involved User:	Guest (becomes Player) – human with browser; no prior knowledge required.
Triggering Event:	Click “Create Account” and submit username.
Superior business process:	BP-2 Lootbox Opening
Goal:	Convert one lootbox into a random stove while keeping drop transparent.
Precondition:	Player logged in; lootboxCount > 0.
Postcondition:	New Stove row inserted; ownership log entry “lootbox”; lootboxCount decremented by 1.
Involved User:	Player – any registered user.
Triggering Event:	Press “Open Lootbox” button.

Field	Description
Use-Case ID	UC-3 List & Buy Stove
Superior business process	BP-4 Stove Listing & Sale
Goal	Enable peer-to-peer exchange with atomic swap and recorded price
Precondition	Seller owns stove; Buyer has ≥ asking price in coins
Postcondition	Ownership transferred; coins moved; median price history updated
Involved User	Seller, Buyer – both registered Players
Triggering Event	Seller lists; Buyer clicks “Buy Now”

Field	Description
Use-Case ID	UC-4 View Price History
Superior business process	Part of BP-4 Stove Listing & Sale
Goal	Let users inspect 30-day median price trend for informed trading
Precondition	Stove type has ≥ 1 past sale (or chart empty)
Postcondition	Line chart rendered showing median values over time
Involved User	Any Player or Guest (read-only)
Triggering Event	Select stove type in marketplace or inventory
Field	**Description**
Use-Case ID	UC-5 Check Ownership Chain

Superior business process	Part of BP-4 Stove Listing & Sale
Goal	Provide transparent provenance for every single stove
Precondition	Valid stoveId entered
Postcondition	Timeline of previous owners displayed
Involved User	Any Player or Guest
Triggering Event	Click “Ownership History” link

2.2.2 GUI to call the use case

Input field	***Valid inputs***
Name of the GUI field	What are valid inputs and what not.
username	3-20 alphanumerics or underscore, unique
Captcha	Correctly solved challenge (Google/hCaptcha/math)
button	Click/tap while lootboxCount > 0
Asking price (seller)	Integer >= 1 coin
Confirm listing	Explicit confirm action (checkbox/OK)
Buy confirmation (buyer)	Click/tap while coins >= asking price
Stove type selector	Any existing StoveType ID or name from dropdown
stoveId	Any existing stove ID (integer, already shown in inventory/market)

Scenario for the standard use (good case)

Step ***User*** ***Activity***

This document doesn't have any headings. To add headings to your Table of Contents, go to Home > Styles

Step	User	Activity
---	---	---
1	Guest	Clicks “Create Account”
2	System	Shows username and Captcha
3	Guest	Enters desired username and solvebs Captcha
4	Guest	Submits form
5	System	Displays “Account created –1000 coins added” and auto-logs the user in

6	Guest	Lands on Dashboard, coins visible, use case ends

Step	User	Activity
---	---	---
1	Player	Clicks “Open Lootbox”
2	System	Shows opening animation and rarity roll
3	System	Displays new stove picture & name; adds it to inventory
4	Player	Closes animation; use case ends

Step	User	Activity
---	---	---
1	Seller	selects stove from inventory and clicks “List for Sale”
2	System	prompts for asking price
3	Seller	enters price ≥ 1 coin and confirms
4	System	locks stove and publishes offer on marketplace
5	Buyer	Browses marketplace, sees offer, clicks “Buy”
6	System	checks Buyer’s balance, performs atomic swap
7	System	shows “Purchase successful” to Buyer; transfers coins to Seller
8	Buyer & Seller	Both receive confirmation; use case ends

Step	User	Activity
---	---	---
1	Player	Selects a stove type (dropdown or tile)
2	System	fetches 30-day median prices
3	System	renders line chart
4	Player	inspects chart; use case ends

Step	User	Activity
---	---	---

1	Player	clicks “Ownership History” on any stove
2	System	queries chain of former owners
3	System	displays timeline with usernames & dates
4	Player	closes overlay; use case ends

Scenarios for non-standard uses (bad cases or work around cases)

Describe all error cases or variations of the use case here.
Aufgabe dieses Abschnittes ist es Fehlerfälle sowie Variationsmöglichkeiten im Ablauf des Use Cases zu beschreiben.

Step	*User*	*Activity*
1	Guest	submits desired username
2	System	checks uniqueness, returns 409 error
3	System	displays “Name taken – try another”
4	User	enters new username and resubmits (loop until success or cancel)

Step	*User*	*Activity*
1	Player	Clicks “open Lootbox”
2	System	checks count = 0
3	System	shows toast “No boxes left – play mini-game to earn more”
4	Player	Closes toast and ends

Step	*User*	*Activity*
1	Buyer	Clicks “open Lootbox”
2	System	checks count = 0
3	System	shows toast “No boxes left – play mini-game to earn more”
4	Buyer	Closes toast and ends

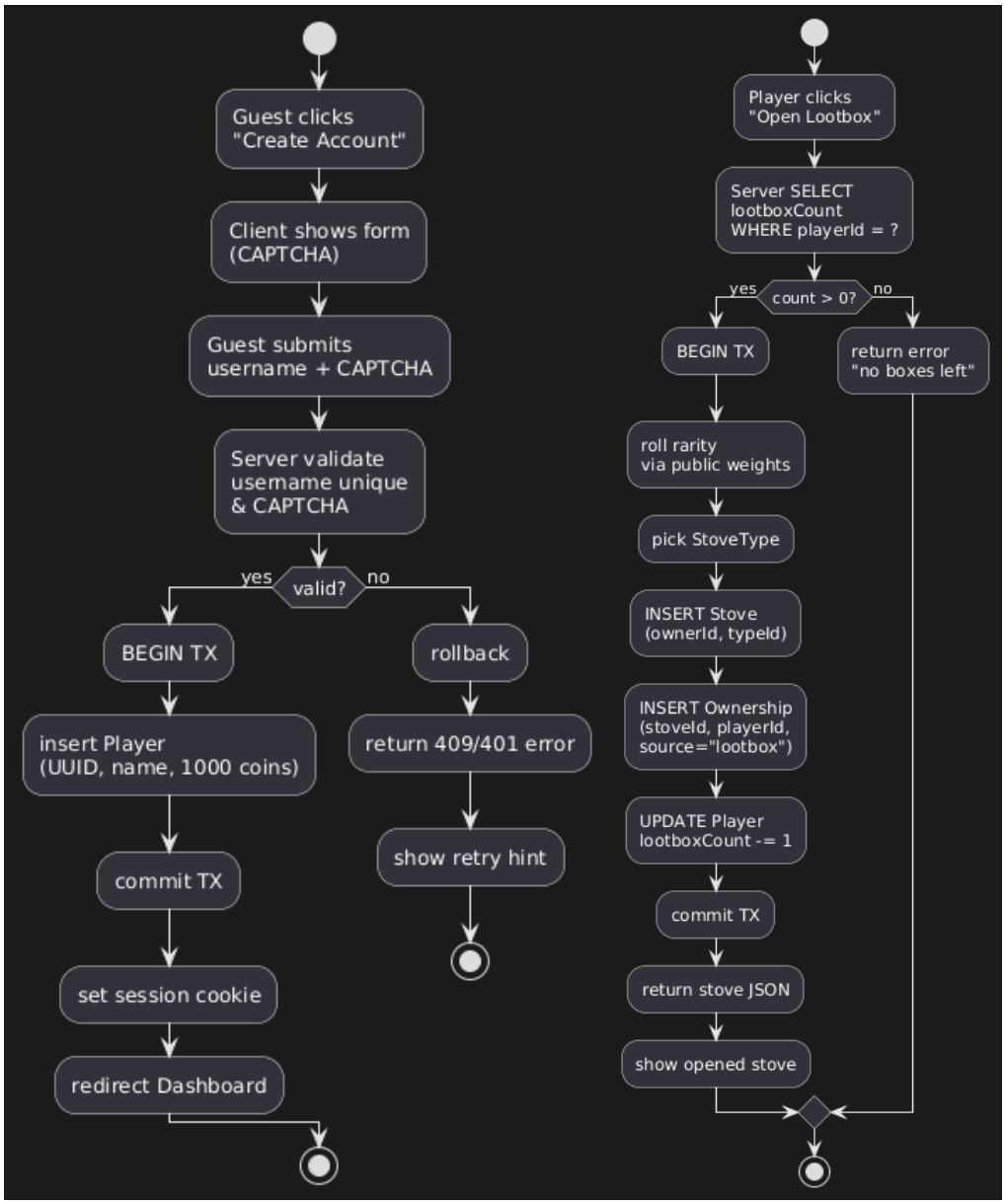
Step	*User*	*Activity*
1	Buyer	clicks “Buy” on listed stove
2	System	Verifies balance < asking price
3	System	Returns error “insufficient funds”

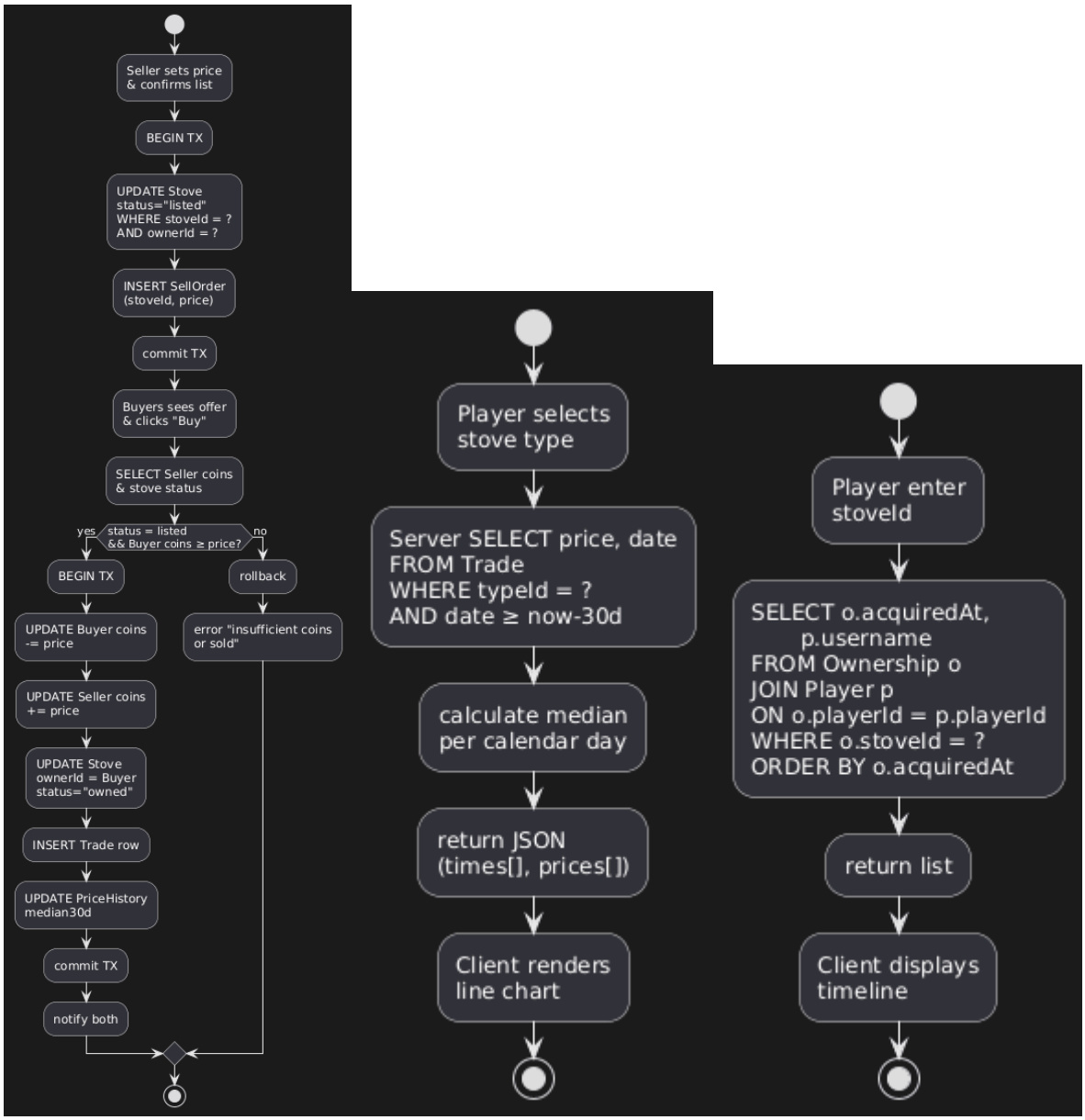
4	Buyer	dismisses error; may choose to earn more coins or abort

Step	***User***	***Activity***
1	Player	Selects a stove type
2	System	Finds zero trades in 30 d
3	System	Displays empty chart + message
4	Player	Acknowledges; use case ends

Step	***User***	***Activity***
1	Player	Enters non existen stove id
2	System	Returns 404
3	System	Offers link back
4	Player	follows link; use case ends

2.2.3 Workflow


Guest clicks "Create Account"
Client shows form (CAPTCHA)
Guest submits username + CAPTCHA
Server validate username unique & CAPTCHA
valid?
yes
no
BEGIN TX
insert Player (UUID, name, 1000 coins)
commit TX
set session cookie
redirect Dashboard
rollback
return 409/401 error
show retry hint
Player clicks "Open Lootbox"
Server SELECT lootboxCount WHERE playerId = ?
count > 0?
yes
no
BEGIN TX
return error "no boxes left"
roll rarity via public weights
pick StoveType
INSERT Stove (ownerId, typeId)
INSERT Ownership (stoveId, playerId, source="lootbox")
UPDATE Player lootboxCount -= 1
commit TX
return stove JSON
show opened stove



2.2.4 Open Points

L

Open Point 1 – Final rarity weights & drop table still need balancing play-tests to avoid inflation or stagnant market.

Open Point 2 – Mini-game faucet payout size and daily cap not yet decided; affects coin sink vs. source balance.

Open Point 3 – Median-price calculation window (30 d vs. 7 d) and update frequency still open; influences price-manipulation detection sensitivity.

Open Point 4 – Admin reset right: who gets the button and how to prevent accidental wipe in production demo?

3 Non-functional Requirements

ID:	NFR_001
Name:	Response-Time Requirement
Type:	EFFIC
Description:	90 % of user-visible actions (open lootbox, place buy order, view chart) finish within 500 ms wall-clock time under 50 concurrent users
Assigned use cases:	UC-2, UC-3, UC-4, UC-5

ID:	NFR_002
Name:	Browser Portability
Type:	MAINT
Description:	Front-end runs unmodified in Chrome ≥ 100, Firefox ≥ 100, Safari ≥ 15; no plug-ins.
Assigned use cases:	All UC

ID:	NFR_003
Name:	Data Integrity on Crash
Type:	SEC
Description:	Every stove transfer or coin movement happens inside a SQLite transaction; committed state must never show negative balances or duplicate stoves after server crash.
Assigned use cases:	UC-2, UC-3

ID:	NFR_004
Name:	No Real-Money Layer
Type:	LEGAL
Description:	System does not accept, store or transmit real currency or crypto; all balances are virtual coins without cash-out option, keeping activity outside gambling law scope.
Assigned use cases:	U C-2, UC-3, UC-4

4 Quantity Structure

Estimated Quantities (per single community server, 3-month cycle)

Master Data

- StoveTypes (static rarity catalogue): 50 rows
- Players: 500 (course + stream audience)
- Lootboxes per player: 100 → 50 000 boxes total

Transaction Data

- Stoves minted: 50 000 (one per opened box)
- Ownership changes (trades): 5 / stove avg → 250 000 trades
- Price-history snapshots: 1 per trade → 250 000 rows
- Chat messages: 10 / player → 5 000 rows
- Mini-game sessions: 20 / player → 10 000 rows

Total DB rows ≈ 560 000 ⇒ fits comfortably in a single SQLite file (< 1 GB).

Peak concurrent users: 50 → UI must support quick search/filter (pagination, auto-complete) but no heavy paging needed.

2 System Architecture and Interfaces

User Interface Layer

- Web Browser (Chrome, Firefox, Safari) – HTTPS/WSS to server; no plug-ins.

Application Layer

- Node.js/Express REST API

- WebSocket endpoint for live price updates

- Session cookie/JWT auth

Data Layer

- Single SQLite file (ember.db) – stores players, stoves, trades, chat

- Daily file-copy backup, no external DB

External Interfaces

- CAPTCHA provider (Google/hCaptcha) – HTTPS call on registration

- Optional SMTP relay – password-reset mails (configurable)No peer systems, no payment gateways, no blockchain.

3 Acceptance Criteria

AC ID	Name of acceptance criterion	Test Step	Expected Behavior
AC_001	Lootbox Drop Transparency	Open 100 lootboxes while capturing drop logs	Relative frequencies per rarity deviate < 5 % from published weights; no stove type skipped
AC_002	Atomic Trade Swap	Run 50 concurrent buy attempts on same listed stove	Exactly one succeeds; balances and owner consistent; no negative balances
AC_003	Price History Accuracy	Execute 30 trades of same stove type at known prices; query history endpoint	Returned 30-day median equals SQL median; response < 500 ms
AC_004	Ownership Chain Completeness	Trade one stove through five accounts; query chain	Timeline lists all five owners in correct order with matching timestamps
AC_005	Registration Performance	JMeter 50 parallel registration calls with unique usernames	95 % response ≤ 500 ms; zero duplicates; CAPTCHA fail < 2 %

List of Abbreviations

Abbreviation	Description
DB	Database
TX	Transaction
UC	Use Case
PK	Primary Key
FK	Foreign Key
UI	User Interface
REST	Representational State Transfer
JWT	JSON Web Token
CAPTCHA	Completely Automated Public Turing Test to Tell Computers and Humans Apart
NFT	Non-Fungible Token (not used herein)
VPS	Virtual Private Server
SQL	Structured Query Language
SQLite	SQL database engine used in this project

4 References

References

1	SQLite Documentation	https://sqlite.org/docs.html
2	Node.js Best Practices	https://github.com/goldbergyoni/nodebestpractices
3	PlantUML Language Reference	https://plantuml.com/guide

Vorgaben zur Prüfung des Dokuments

(Quelle: http://www.fh-augsburg.de/informatik/vorlesungen/se1t/script/definition/review.html)

Review nach der Definitionsphase

- Haben die Teilnehmer das Problem gleichermaßen verstanden?
- Wie wurde das Problem erfasst? Wer wurde befragt?
- Welche Teile sind zur Wiederverwendung geeignet?
- Wo ist das Programm besonders komplex?
- Sind die Anforderungen des Auftraggebers vollständig erfasst worden?
- Wurde das System mit dem Auftraggeber besprochen?
- Wurden Kompromisse zwischen Auftraggeber und Entwickler ausgehandelt ? Welche, Begründung ?
- Ist das Problem Ablauf- oder Datenorientiert?
- Welche Definitionsmethoden wurden für das Pflichtenheft verwendet? Waren die Methoden geeignet ?

- Hat der Auftraggeber den Leistungsumfang des Systems völlig verstanden?
- Wurden technische und wirtschaftliche Restriktionen vollständig berücksichtigt?
- Wurden technische und wirtschaftliche Risiken realistisch eingeschätzt?
- Sind die notwendigen Ressourcen für die Entwicklungsarbeit vorhanden?
- Wer ist der verantwortliche Ansprechpartner nach Ende der Definitionsphase?
- Sind Abnahmekriterien aufgestellt?
- Sind Copyrights geklärt?
- Was kann bzgl. Kommunikation zwischen Auftraggeber und Entwickler in Zukunft verbessert werden ?
- Bestehen Zielkonflikte aufgrund von Qualitätsmerkmalen?
- Haben Entwickler die Machbarkeit bestätigt?
- Wurden, wo nötig, Prioritäten gesetzt?
- Gibt es ungeklärte Einwände des Auftraggebers?
- Kann das Entwicklungsrisiko weiter vermindert werden?

Abschließende Frage:
- Soll das Projekt weiter durchgeführt werden?

Mögliche Antworten:
- ja, alles läuft wie vorgesehen, geringe Abweichungen sind notiert.
- ja, aber die letzte Phase muss mit einem Aufwand von xx% wiederholt werden; es entsteht eine entsprechende Verzögerung, die jedoch noch im Rahmen der Wirtschaftlichkeit bleibt.
- nein, es sind unerwartete Schwierigkeiten aufgetreten, die eine wirtschaftliche Durchführung des Projektes nicht mehr erwarten lassen.