Design Specifications (G3)

Anshul Sangrame, Rajiv Chitale, Rishit D, Suryaansh Jain

Overview

Software

Our software aims to provide an end-to-end solution for customers participating in online auctions. We provide personalized interfaces for all users taking into account their preferences and auction history. We allow prospective auctioneers to declare items and schedule auctions at their convenience. Our software provides for multiple bidders to join auction rooms, leave auction rooms and bid for items of interest.

Design

This document encapsulates our data-flow-diagrams, their abstract inputs and outputs along with the appropriate structured charts and their comprising modules. We also specify each module type and its predicted complexity in the system in terms of cohesiveness and coupling. This document highlight possible bottlenecks and error-prone modules with intuitive reasoning and mathematical analysis. We also consider the tentative lines-of-code (LoC) for every input, transform and output subsystems./ We summarize our software through an interface sketch comprising of relevant classes, attributes, methods and inheritances.

Data Flow Diagrams

We elucidate the data-flow of our software through the following diagrams. To simplify our data-flow and modularize our application we effectively split our software into three very loosely coupled systems - authentication, general utilities and auction-room management. We have also highlighted the most abstract inputs and outputs for each of these subsystems in their reprective diagrams.

<Insert Diagrams Here!!>

Structured Charts

We now list sequentially modules formed from the aforementioned data-flow duagrams via factoring at different levels and the final structured chart with all modules combined.

First Level Factored Modules

Factored Input Modules

Factored Output Modules

Factored Transform Modules

Final Structured Chart

Design Analysis

List of Modules

We list our final factored modules in the table below with corresponding type of module and cohesion, and its estimated size.

Module Name	Module Type	Cohesion Type	Estimated Size
Bid Main	Composite	Functional	
Get Bid Price	Input	Functional	
Get Auction ID (URL)	Input	Functional	
Positive Bid	Transform	Logical	
Item Main	Composite	Functional	
Get Item Name	Input	Functional	
Get Item Tag	Input	Functional	
Get Item Image	Input	Functional	
Get Item St Price	Input	Functional	
Get Auth Token	Input	Functional	
Get Auction ID (Room)	Input	Functional	
Get Current Time	Input	Temporal	
Get End Leaderboard (ab)	Composite	Temporal/Sequential	
Query for End Auction	Transform	Logical/Temporal	
Exec Query End Auction	Transform	Logical/Temporal/Fnl	
Add Item (DB)	Output	Functional	
Get Highest Bid	Composite	Communicational/Fnl	
Query for Highest Bid	Transform	Functional	
Exec Query Highest Bid	Transform	Functional	
Update Leaderboard	Composite	Logical/Functional	
Check Greater User	Transform	Logical/Functional	
Update Leaderboard Bids	Output	Functional	
Get Username	Composite	Functional	
Get Encryption Key	Input	Functional	
Decrypt Token	Transform	Functional	
Update Interest	Output	Functional	
Get Leaderboard	Composite	Functional	
Query for Leaderboard	Transform	Functional	
Exec Leaderboard Query	Transform	Functional	
Get Auction Item List	Composite	Functional	
Query for Item List	Transform	Functional	
Exec Query for Item List	Transform	Functional	
Update Auction History	Output	Functional	
Display for Auction Room	Coordination	Functional	
Display Item List	Coordination	Functional/Temporal	
Display Item	Output	Functional	
Display Leaderboard	Output	Functional	
Notify Winner	Composite	Functional	
Email Template	Transform	Functional	
Send Email	Output	Functional	
Store Auction End	Output	Functional	

Module Justifications

In general we can observe a few trends that can characterized as follows:
1. **Query - Get Information - Execute Query Triads**: These subsystems consist of three modules - one transform module that consists forming the query, one composite module that handles getting information for the main module and one transform module that is responsible for executing the query. In most of these cases, all three modules are significantly coupled as all three modules are interconnected and dependent on each other but we can argue that these modules are not tightly coupled as they connections are weak and are only limited to a fan-in and fan-out of 1. Again it can be noted that all three modules primarily exhibit functional cohesiveness with the central composite module also serving secondarily as a communication module.

2. **Central Continuous Forms**: These subsystems consist of a central module that continuously expects actions and information and passes on information from its subordinate modules. The central module is a composite module with functional cohesiveness as it passes on information towards the main module and continously loops in anticipation.

The suboordinate modules are mostly functionally cohesive for information and are logically cohesive when the module is triggered by an action. Again, these modules have significant coupling, especially at the central module as it takes in multiple paramaters. But again, coupling is significantly reduced due to clearly defined entry points and parameters which are passed on.

3. **Updates**: These subsystems consist of a central module and two suboordinate modules. The central module fetches information (sometimes conditionally) from a transform module and instructs another output module to update/execute certain instructions. This is similar to the Query Triad except that the final subordinate module is not obligated to inform the central module. Again most such modules are functionally cohesive except when the first subordinate module is action-dependent (in which case it is logically cohesive). Low coupling is observed in such subsystems as number of interconnections are small and weak due to effective parameterization.

We now provide a brief justification for the type of cohesion and coupling expected in the above module in regards to our software. We also refer to the previous trends whenever applicable:

- **Bid Main**: This module is responsible for the compiling the bid object parameters from the user provided the user has positively responded to the bidding option. This serves as a central composite module as referred to in (2). The following 3 modules serve as subordinates to this module.

- **Get Bid Price**: This module is repsonsible for receiving bidding price and answers to the **Bid Main** module.

- **Get Auction ID (URL)**: This module is responsible for receiving the corresponding auction_id to the **Bid Main** module.

- **Positive Bid**: This module is logically cohesive as it checks if the present bidding value is positive with respect to the present highest bid. This again serves as a subordinate to the **Bid Main** module.\
- **Item Main**: Similar to the previous subsystem, this serves as the central module of the subsystem described in (2) with the following 4 modules serving as subordinates. This serves as a composite module that collects item information to be added in the auction room.

- **Get Item Name**: This module fetches the item name for **Item Main**.

- **Get Item Tags**: This module fetches relevant tags for the item for **Item Main**.

- **Get Item Image**: This module fetches the associated image for the item for **Item Main**.

- **Get Item Starting Price**: This module fetches the item's base price for **Item Main**.

- **Get Auth Token**: This module authenticates a user entering the auction room via the user's current authentication token. This is a simple input module which is functionally cohesive and has low coupling with the main module through a single parameter - auth_token.\
- **Get Auction ID (Room)**: This module fetched the auction room for a user joining the room through a link. This module is an input module with functional cohesiveness and low coupling with the main module.\
- **Get Current Time**: This module fetches current time for all time sensitive operations with main module. This module is an input module with temporal cohesiveness. It can argued that this module has high coupling with the main modules and associated time-dependent modules. This is intuitive as cohesiveness and coupling are oppositely correlated.\
- **Get End Leaderboard (end initiated)**: This module is triggered when the auctioneer ends the auction before the specified time. Note that this serves as the central module of the Query Triad subsystem. Although we would expect this module to be functionally cohesive, we can argue that this module leans more towards temporal cohesiveness due to its massive dependence on time. Again by correlation, we see that this tightly coupled with the main module

and the time-fetching module.

- **Query for End Auction**: This serves as the transform module that forms the query when the auction ends. Refer the Query Triad subsystem for further information.

- **Execute Query for End Auction**: This serves as the transform module that executes the above query.\
- **Add Item to Database**: This module adds an item to the common database for items for further recommendations and preferences. This is a simple input module with functional cohesiveness and has low coupling with the main module.\
- **Get Highest Bid**: This module again serves as the central module of the Query Triad subsystem. Please refer to subsystem (1) for further details. The following two modules are transform modules and form and execute the necessary query.

- **Query for Highest Bid**: This module forms the query to extract the highest bid for an auction.

- **Execute Query for Highest Bid**: This module executes the above query.\
- **Update Leaderboard for Bid**: This module serves as the central module for the subsystem described in (3). We also comment the that

- **Check User with Greater Bid**:

- **Update Leaderboard with Bid**: \
- **Get Username**:

- **Get Encryption Key**:

- **Decrypt Token**: \
- **Update Interest**: \
- **Get Leaderboard for View**:

- **Query for Leaderboard**:

- **Execute Query for Leaderboard**: \
- **Get Auction Item List**:

- **Query for Item List**:

- **Execute Query for Item List**: \
- **Update Auction History**: \
- **Display for Auction Room**:

- **Display Item List**:

- **Display Item**:

- **Display Leaderboard**: \

- **Notify Winner**:
- **Email Template**:
- **Send Email**: \
- **Store Auction End**: \