Data Formatting

Customer:

- Email: max 36 chars, XXXXXX@XXXX.XXX
- Password: max 20 chars Example : qwerty
- Name: First Name: max 25 chars; Last Name: max 40 chars
- Home Phone: “(ddd) ddd-dddd”
- Work Phone: “(ddd) ddd-dddd”
- Address: max 50 chars

Clerk:

- Login Key: max 16 chars
- Password: max 20 chars
- Name: First Name: max 25 chars; Last Name: max 40 chars

Reservation:

- Reservation Number: int
- Date: Start Date: ‘MM-DD-YYYY’; End Date: ‘MM-DD-YYYY’
- Total Rental Price: float, round to two decimal places
- Total Deposit: float, round to two decimal places

Picks up:

- Credit Card Details: Card Number: ‘dddd-dddd-dddd-dddd’; Expiration Date: ‘MM/YY’

Tool:

- Tool ID: int
- Abbreviated Description: max 25 chars
- Full Description: max 300 chars
- Original Purchase Price: float, round to two decimal places
- Daily Rental Price: float, round to two decimal places
- Deposit Amount: float, round to two decimal places

Power Tool:

- Accessories: multi-value with max 25 chars, each

Service Order:

- Date: Starting Date: ‘MM-DD-YYYY’; Ending Date: ‘MM-DD-YYYY’
- Estimated Repair Cost: float, round to two decimal places

Constraints

- End date should be after start date.
- Original purchase price, daily rental price and deposit amount of a tool should be positive.
- A tool is available for rental only if it isn’t reserved or being repaired from start date to end date.
- A user is allowed to rent up to 50 tools per reservation.
- The total rental price of a reservation is calculated as the sum of the rental prices for all tools rented, multiplied by the number of days over which they are rented.
- The total deposit price of a reservation is the sum of the deposits required for each individual tool.
- A service order cannot be placed over an existing reservation.
- Estimated repair cost should be positive.

Task Decomposition and Abstract Code

Login

Task Decomposition

- Lookup personal information for a customer or a clerk
- One schema construct is needed
- No enabling condition
- Consistent frequency
- Read only
- Consistency is not essential
- Mother task is not needed

Abstract Code

While the **Enter** button is not pushed, do nothing;
When the **Enter** button is pushed, check if all fields (Login, Password, User type) are filled and do the following:

{If empty field found: display error message (“All the fields are required”);
If all filled, do the following:
    {If user type is **Customer**: Find the current customer using username;
        {If not found: Direct to create profile page;
        If found: Check if password matches username;
            {If match: login successful, direct to customer main menu;
            If not match: display error message ("Login failed, try again")}};
    If user type is **Clerk**: Find current clerk using clerk’s login key;
        {If clerk not found: Display error message (“Login key not found”)
        If clerk found: Check if password matches clerk login key
            {If match: login successful, direct to clerk main menu;
            If not match: display error message ("login failed, try again")}}}};

Create profile

Task Decomposition

- Lookup personal information for a user in case of duplicate; Insert personal information of a new user
- One schema construct
- No enabling condition
- Read and insert
- Consistent frequency

- Consistency is essential
- Mother task is needed

Abstract Code

While the **Submit** button is not pushed, do nothing;
When the **Submit** button is pushed, then do the following:
{If empty field found: display error message (“All the fields are required”);
If all fields filled: Insert a new customer; direct to login interface};

View profile

Task Decomposition

- Lookup customer personal information; Lookup reservation
- Several schema constructs are needed
- Enabled by a customer’s login and view profile request
- Consistent frequency
- Read only
- Consistency is not essential
- Mother task is needed

Abstract Code

Find current **Customer** using customer email address;
Display **Customer** email address, name, home phone, work phone, address;
Find all reservations for the current **Customer**
    {Sort all reservations from most recent to oldest;
    Start from the most recent, for each found reservation
        {Display reservation number, tools, start date, end date, total rental price, total deposit, pick-up clerk, drop-off clerk};

Check tool availability

Task Decomposition

- Lookup type of tools; Lookup reservations and service orders of tools
- Three schema constructs
- Enabled by a customer’s login and check tool availability request
- Read only
- Consistent frequency
- Consistency is essential
- Mother task is needed

Abstract Code

While the **Submit** button is not pushed, do nothing;
When the **Submit** button is pushed, then do the following:

{If empty field found: display error message (“All the fields are required”);
If all fields filled: Find all tools in inventory that match the provided tool type
    {Find reservations and service orders for all previously found tools using the tool ID;
    {Find all tools in previously found tools table that are not associated with a
    service order or reservation during the specified dates:
        {Display tool number, abbreviated description, deposit, price/day}}};
If a tool ID is entered and **View Details** button is pushed:
    {Find the tool using tool ID;
    Display tool ID, abbreviated description, full description, original purchase price, daily rental price, deposit amount}};

Make a reservation

Task Decomposition

- Lookup list of tool types; Lookup rental price and deposit amount of tools; Insert a new reservation
- Several schema constructs
- Enabled by a customer’s login and make reservation request
- Read and insert
- Different frequencies
- Consistency is essential
- Mother task is needed

Abstract Code

Populate type of tool dropdown;
If a tool type is selected:
{Check tool availability;
Populate available tool dropdown, for each tool in dropdown
    {Display tool ID, abbreviated description, daily rental price}};
While no button is pushed, do nothing;
When a button is pushed, then do the following:
{If **Add More Tools**: display the form with room for a new tool;
If **Remove Last Tool**: remove last tool;
If **Calculate Total**:
    {For each selected tool, find daily rental price and deposit;
    Calculate total rental price and total deposit required;

Display the abbreviate description of selected tools, start date, end date, total rental price and total deposit;
While no button is pushed, do nothing;
When the **Submit** button is pushed:
{Insert reservation;
Display tools, start date, end date, total rental price, total deposit;
When **Back to Main Menu** button is pushed, direct to main menu};
When **Reset** button is pushed: display an empty make reservation form}};

Pick-up

Task Decomposition

- Lookup reservation by reservation number; Update credit card info in reservation; Lookup tool detail description; Update reservation pick up clerk;
- Several schema constructs
- Enabled by a clerk's login and pick-up request
- Read and update
- Different frequencies
- Consistency is essential
- Mother task is needed

Abstract Code

When **Pick-Up** button is pushed and the reservation number is entered:
{Find the reservation using reservation number;
Display reservation number, tools required, deposit required, estimated cost;
If a tool ID is entered and **View Details** button is pushed:
{Find the tool using tool ID;
Display tool ID, abbreviated description, full description, original purchase price, daily rental price, deposit amount};
If **Complete Pick-Up** button is pushed:
{Update credit card number and expiration date for the reservation;
Update the pick-up clerk of the reservation;
Display the rental contract with reservation number, pick-up clerk, customer's name, credit card, start date, end date, a list of tools rented, total deposit and total rental price of the reservation}};

Drop-off

Task Decomposition

- Lookup reservation by reservation number; Update reservation drop off clerk; Lookup tool detail description

- Several schema constructs
- Enabled by a clerk's login and drop-off request
- Read and update
- Different frequencies
- Consistency is essential
- Mother task is needed

Abstract Code

When **Drop-Off** button is pushed and the reservation number is entered:
{Find the reservation using reservation number;
Display reservation number, tools required, deposit required, estimated cost;
If a tool ID is entered and **View Details** button is pushed:
{Find the tool using tool ID;
Display tool ID, abbreviated description, full description, original purchase price, daily rental price, deposit amount};
If **Complete Drop-off** button is pushed:
{Update the drop-off clerk of the reservation;
Display the receipt with reservation number, drop-off clerk, customer's name, credit card, start date, end date, total rental price, deposit held and total outstanding balance of the reservation}};

Add tool

Task Decomposition

- Lookup tool type; Insert a new tool
- One schema construction
- Enabled by a clerk's login and add new tool request
- Read and insert
- Different frequencies
- Consistency is essential
- Mother task is needed

Abstract Code

When **Add New Tool** button is pushed:
{Display the Add New Tool interface;
Populate tool type dropdown;
If the tool type is "Power Tool":
{When **Add Accessories** button is pushed: Display the add accessories interface for the clerk to add accessories}
If **Submit New Tool** button is pushed: Insert a new tool}

Sell tool

Task Decomposition

- Lookup original purchase price of a tool; Insert a tool in tool for sale table; Delete a tool in tool table
- Several schema constructs
- Enabled by a clerk's login and sell tool request
- Read, delete and insert
- Different frequencies
- Consistency is essential
- Mother task is needed

Abstract Code

When **Sell Tool** button is pushed:
    {Display the Sell Tool interface;
    When the tool ID is entered:
        {Calculate the sale price (original price divided by 2);
        Display abbreviate description and sale price of the tool;
        Insert a new tool in tools for sale table, the informations of the new tool is the same as the selected tool;
        Delete the selected tool in tool table}};

Hold tool for repair

Task Decomposition

- Lookup of reservations; Insert a new service order;
- Several schema constructs
- Enabled by a clerk's login and service order request
- Read and insert
- Different frequencies
- Consistency is essential
- Mother task is needed

Abstract Code

When **Service Order** is pushed:
    {Display the service order request interface;
    If **Submit** button is pushed:
        {Find reservations using tool ID
        If the tool has been reserved during specified dates:
            {Display "This tool has been reserved during specified dates.";
            Direct to service order request interface};

If the tool has not been rented: Insert a new service order}};

Generate reports

Task Decomposition

- Lookup tool; Lookup reservation; Lookup service order requests
- Several schema constructs
- Enabled by a clerk's login and generate reports request
- Read
- Different frequencies
- Consistency is not essential
- Mother task is needed

Abstract Code

When **Generate Reports** is pushed:
For each tool in inventory:

{Find related reservations using tool ID, sum up the number of days that the tool has been rented;
Calculate the rental profit, which is the rental price multiplied by the number of days the tool has been rented;
Find related service order requests using tool ID, sum up the total cost of services;
Calculate the cost of the tool, which is original price of the tool plus the cost of any service orders made on the tool;
Calculate the total profit, which is the rental profit minus the cost of the tool;
Display tool ID, abbreviated description, rental profit, cost of tool, and total profit, sorted by total profit};

Find the reservations made in last month, for each rental customer who rented a tool over the last month:

{Sort the selected reservations by email address;
Count the number of rentals corresponding to each email address;
Display names of rental customers in last month, email address, and number of rentals (ordered first by number of tools rented, and then last name)};

Select all the reservations handled during last month:

{Group the number of pick-ups and the number of drop-offs by clerk login key;
Find each clerk’s name by clerk login key;
Display the list the clerk name, the number of pick-ups handled this month, the number of drop-offs handled this month, corresponding to each login key, ordered by the total number of pick-ups and drop offs};