**Login** Shailesh

Abstract Code

* User enters email/nickname($UserID) and password($Password) and clicks on the Login button.
* User credentials check task

|  |
| --- |
| SELECT email from `User` where (User.email=’$UserID’ or User.nickname=’$UserID’) and User.password=’$Password’; |

* If an empty row is returned through the query, display an error message in the screen stating as an invalid login credentials.
* If a user row is returned, store the returned email address in variable ’$UserID’ and navigate the user to Main Menu.

**Registration** Shailesh

Abstract Code

* User enters email($UserID), password($Password), nickname($NickName), firstname($FirstName), lastname($LastName) and postalcode($PostalCode) and clicks on the Register button.
* New user registration credentials check task

|  |
| --- |
| INSERT INTO `User` (email, password, nickname, first\_name, last\_name, FK\_Address\_PostalCode) VALUES ($UserID, $Password), $NickName, $FirstName, $LastName, $PostalCode)); |

* If an error is returned from the query, display an error message in the screen stating as an invalid registration credentials.
* If a user row is returned, store the returned email address in variable ’$UserID’ and navigate the user to Main Menu.

**Main Menu** Shailesh

Abstract Code

* Fetch and display firstname, lastname and nickname of user using $UserID.
* User Details fetch task

|  |
| --- |
| SELECT first\_name, last\_name, nickname from `User` where (User.email=’$UserID’ or User.nickname=’$UserID’); |

* Display firstname, lastname and nickname of the user in the main menu.
* Fetch average response time for accepting/rejecting a trade by the user.
* Fetch average response time task

|  |
| --- |
| Select ROUND(avg(TIMESTAMPDIFF(DAY,accept\_reject\_date,NOW())),1) from Trade Inner Join  (  Select Item.item\_number, ItemJoin.FK\_User\_Email from Item  NATURAL JOIN (  Select BoardGame.FK\_Item\_Item\_Number, BoardGame.FK\_User\_Email from BoardGame  UNION  Select PlayingCardGame.FK\_Item\_Item\_Number, PlayingCardGame.FK\_User\_Email from PlayingCardGame  UNION  Select CollectibleCardGame.FK\_Item\_Item\_Number, CollectibleCardGame.FK\_User\_Email from CollectibleCardGame  UNION  Select ComputerGame.FK\_Item\_Item\_Number, ComputerGame.FK\_User\_Email from ComputerGame  UNION  Select VideoGame.FK\_Item\_Item\_Number, VideoGame.FK\_User\_Email from VideoGame  ) AS ItemJoin where '$UserID' = ItemJoin.FK\_User\_Email  )  AS TradeJoin ON Trade.FK\_CounterParty\_Item\_Item\_Number= TradeJoin.FK\_Item\_Item\_Number  where Trade.trade\_status='Accepted' or Trade.trade\_status='Rejected'  GROUP BY TradeJoin.FK\_User\_Email; |

* If nothing gets returned, display None with black text.
* If data is returned, display text with corresponding color.
* Fetch number of unaccepted trades by the user.
* Fetch number of unaccepted trades task

|  |
| --- |
| Select count(\*) from Trade Inner Join  (  Select Item.item\_number, ItemJoin.FK\_User\_Email from Item  NATURAL JOIN (  Select BoardGame.FK\_Item\_Item\_Number, BoardGame.FK\_User\_Email from BoardGame  UNION  Select PlayingCardGame.FK\_Item\_Item\_Number, PlayingCardGame.FK\_User\_Email from PlayingCardGame  UNION  Select CollectibleCardGame.FK\_Item\_Item\_Number, CollectibleCardGame.FK\_User\_Email from CollectibleCardGame  UNION  Select ComputerGame.FK\_Item\_Item\_Number, ComputerGame.FK\_User\_Email from ComputerGame  UNION  Select VideoGame.FK\_Item\_Item\_Number, VideoGame.FK\_User\_Email from VideoGame  ) AS ItemJoin where '$UserID' = ItemJoin.FK\_User\_Email  )  AS TradeJoin ON Trade.FK\_CounterParty\_Item\_Item\_Number= TradeJoin.FK\_Item\_Item\_Number  where Trade.trade\_status='Proposed'  GROUP BY TradeJoin.FK\_User\_Email; |

* If nothing gets returned, display 0 without any links.
* If greater than 0 but less than 2, provide link to Accept/Reject form.
* If greater or equal to 2, change color to red(underlined) and provide link to Accept/Reject form.
* Fetch current trade rank of the user.
* Fetch current trade rank task

|  |
| --- |
| Select count(\*) from Trade Inner Join  (  Select Item.item\_number, ItemJoin.FK\_User\_Email from Item  NATURAL JOIN (  Select BoardGame.FK\_Item\_Item\_Number, BoardGame.FK\_User\_Email from BoardGame  UNION  Select PlayingCardGame.FK\_Item\_Item\_Number, PlayingCardGame.FK\_User\_Email from PlayingCardGame  UNION  Select CollectibleCardGame.FK\_Item\_Item\_Number, CollectibleCardGame.FK\_User\_Email from CollectibleCardGame  UNION  Select ComputerGame.FK\_Item\_Item\_Number, ComputerGame.FK\_User\_Email from ComputerGame  UNION  Select VideoGame.FK\_Item\_Item\_Number, VideoGame.FK\_User\_Email from VideoGame  ) AS ItemJoin where '$UserID' = ItemJoin.FK\_User\_Email  )  AS TradeJoin ON Trade.FK\_CounterParty\_Item\_Item\_Number= TradeJoin.FK\_Item\_Item\_Number  Inner Join ON Trade.FK\_Proposer\_Item\_Item\_Number= TradeJoin.FK\_Item\_Item\_Number  where Trade.trade\_status='Accepted'  GROUP BY TradeJoin.FK\_User\_Email; |

* If nothing gets returned, display None.
* Display the badge according to the count returned.
* User clicks on the logout Button, User is directed to the login screen.
* User clicks on the List Item Button, User is directed to Listing Items Page.
* User clicks on the My Items Button, User is directed to the My Items Page.
* User clicks on the Search Items Button, User is directed to the Search Items Page.
* User clicks on the Trade History Button, User is directed to the Trade History Page.

**Propose Trade**

Abstract Code

* If the counterparty distance exceeds 100.0, a warning message with the distance displayed at the top should be displayed.

|  |
| --- |
| SELECT proposer\_item\_number, counter\_party\_item\_number, (3958.75 \* 2 \* POWER(ATAN(SQRT((POWER(SIN((offered\_address.Latitude - my\_address.Latitude) / 2), 2) + COS(my\_address.Latitude) \* COS(offered\_address.Latitude) \* POWER(SIN((offered\_address.Longitude - my\_address.Longitude) / 2), 2))), SQRT(1 - (POWER(SIN((offered\_address.Latitude - my\_address.Latitude) / 2), 2) + COS(my\_address.Latitude) \* COS(offered\_address.Latitude) \* POWER(SIN((offered\_address.Longitude - my\_address.Longitude) / 2), 2)))), 2)) as distance  FROM Trade INNER JOIN  (SELECT item\_number, email FROM BoardGame UNION  SELECT item\_number, email FROM CollectibleCardGame UNION  SELECT item\_number, email FROM ComputerGame UNION  SELECT item\_number, email FROM PlayingCardGame UNION  SELECT item\_number, email FROM VideoGame  ) AS offered\_item ON proposer\_item\_number = offered\_item.item\_number INNER JOIN  tradeplazauser as offered\_user ON offered\_user.email = offered\_item.email INNER JOIN  address as offered\_address ON offered\_user.postal\_code = offered\_address.postal\_code INNER JOIN  (SELECT item\_number, email FROM BoardGame UNION  SELECT item\_number, email FROM CollectibleCardGame UNION  SELECT item\_number, email FROM ComputerGame UNION  SELECT item\_number, email FROM PlayingCardGame UNION  SELECT item\_number, email FROM VideoGame  ) AS my\_item ON counter\_party\_item\_number = my\_item.item\_number INNER JOIN  tradeplazauser as my\_user ON my\_user.email = my\_item.email INNER JOIN  address as my\_address ON my\_user.postal\_code = my\_address.postal\_code  WHERE counter\_party\_item\_number IS NULL |

* Only items that are available for trading must be displayed (self).
* Provide an appropriate mechanism for selecting the desired item.
* Should be displayed Number of items, game type, title, and condition
* Items should by ItemNumber.

|  |
| --- |
| SELECT item\_number, title, game\_condition, game\_type  FROM (  SELECT item\_number, title, game\_condition, 'Board Game' AS game\_type FROM BoardGame UNION  SELECT item\_number, title, game\_condition, 'Playing Cards' as game\_type FROM PlayingCardGame UNION  SELECT item\_number, title, game\_condition, 'Collectible Card Game' AS game\_type FROM CollectibleCardGame UNION  SELECT item\_number, title, game\_condition, 'Video Game' AS game\_type FROM VideoGame UNION  SELECT item\_number, title, game\_condition, 'Computer Game' AS game\_type FROM ComputerGame  ) as all\_games NATURAL JOIN tradeplazauser  WHERE (email = "$UserId" OR nickname = "nickname\_100") AND  item\_number NOT IN (  SELECT proposer\_item\_number as item\_number FROM Trade WHERE trade\_status = "ACCEPT" OR trade\_status = "REJECT"  )  ORDER BY item\_number ASC; |

* Once the proposed item has been selected, the user must be given a confirm
* Button to confirm the proposal.
* Once confirmed, the proposed date is saved/written into the Trade table.

|  |
| --- |
| UPDATE Trade SET counter\_party\_item\_number = "$CounterItemNumber" WHERE proposer\_item\_number = "$ProposeItemNumber" AND counter\_party\_item\_number IS NULL |

* The user should be able to return to the main menu after receiving a confirmation message

**Accept/Reject Trade**

Abstract Code

* Results of a Search
* List proposed trades in which the user is the counterparty and can accept or reject a trade. For each proposal show the date proposed, the desired item's title, the proposer's nickname, their rank, distance from the user (rounded to the hundredths), and the proposed item title, all ordered by proposal date.

|  |
| --- |
| SELECT proposed\_date, my\_item.title, my\_item.item\_number, my\_user.nickname, offered\_user.nickname, offered\_item.title, offered\_item.item\_number, (3958.75 \* 2 \* POWER(ATAN(SQRT((POWER(SIN((offered\_address.Latitude - my\_address.Latitude) / 2), 2) + COS(my\_address.Latitude) \* COS(offered\_address.Latitude) \* POWER(SIN((offered\_address.Longitude - my\_address.Longitude) / 2), 2))), SQRT(1 - (POWER(SIN((offered\_address.Latitude - my\_address.Latitude) / 2), 2) + COS(my\_address.Latitude) \* COS(offered\_address.Latitude) \* POWER(SIN((offered\_address.Longitude - my\_address.Longitude) / 2), 2)))), 2)) as distance  FROM Trade INNER JOIN  (SELECT item\_number, email, title FROM BoardGame UNION  SELECT item\_number, email, title FROM CollectibleCardGame UNION  SELECT item\_number, email, title FROM ComputerGame UNION  SELECT item\_number, email, title FROM PlayingCardGame UNION  SELECT item\_number, email, title FROM VideoGame  ) AS offered\_item ON proposer\_item\_number = offered\_item.item\_number INNER JOIN  tradeplazauser as offered\_user ON offered\_user.email = offered\_item.email INNER JOIN  address as offered\_address ON offered\_user.postal\_code = offered\_address.postal\_code INNER JOIN  (SELECT item\_number, email, title FROM BoardGame UNION  SELECT item\_number, email, title FROM CollectibleCardGame UNION  SELECT item\_number, email, title FROM ComputerGame UNION  SELECT item\_number, email, title FROM PlayingCardGame UNION  SELECT item\_number, email, title FROM VideoGame  ) AS my\_item ON counter\_party\_item\_number = my\_item.item\_number INNER JOIN  tradeplazauser as my\_user ON my\_user.email = my\_item.email INNER JOIN  address as my\_address ON my\_user.postal\_code = my\_address.postal\_code  WHERE my\_user.email = "$UserId" or my\_user.nickname = "nickname\_1" AND trade\_status = "PENDING" |

* Both item fields should contain a link to the item's detail page.
* A mechanism for accepting/rejecting should be provided: Accept, Reject
* If the trade is accepted, query the USER table and display a dialog containing the proposer's email address and first name.
* The trade's acceptance or rejection date is recorded in the database as part of the trade table
* If the proposed trade is turned down, a new trade for the same proposed item and the same desired item cannot be proposed.

|  |
| --- |
| UPDATE Trade SET trade\_status = "$ACCEPT/REJECT", accept\_reject\_date = NOW() WHERE proposer\_item\_number = "$ProposeItemNumber" AND counter\_party\_item\_number = "$CounterItemNumber" |

* Accepting a trade removes the item from the listing (Search and Display Search Results), and the user should be returned to the main menu if no more trades need to be accepted or rejected.

**View Trade History** Chris G  
Abstract Code

* User clicked on ***View Trade History*** from **Main Menu**
* **Trade Summary Task**
  + Find current User using global variable *User.email*, from login

|  |
| --- |
| SELECT Email FROM `User` WHERE User.email=’UserID’; |

* + Display **Trade Summary** form at the top of **Trade History** form, provides statistics when current User is Proposer or Counterparty, show Total appearance, Accepted count, Rejected count, and Reject rate

|  |
| --- |
| SELECT “My role”  COUNT(\*) AS Total,  SUM( IF (trade\_status=”Accepted”, 1, 0)) AS Accepted,  SUM( IF (trade\_status =”Rejected”, 1, 0)) AS Rejected,  FORMAT( Rejected / Total , ‘P1’) AS “Rejected %”  FROM (  SELECT trade\_status,  IF(P.OwerID=’$UserID’, “Proposer”, “Counterparty”) AS ‘My role’,  FROM Trade AS T  INNER JOIN (  SELECT FK\_Item\_Item\_Number, FK\_User\_Email from BoardGame  UNION  SELECT FK\_Item\_Item\_Number, FK\_User\_Email from PlayingCardGame  UNION  SELECT FK\_Item\_Item\_Number, FK\_User\_Email from CollectibleCardGame  UNION  SELECT FK\_Item\_Item\_Number, FK\_User\_Email from ComputerGame  UNION  SELECT FK\_Item\_Item\_Number, FK\_User\_Email from VideoGame  ) AS PI ON T.FK\_Proposer\_Item\_Item\_number=PI.FK\_Item\_Item\_Number  INNER JOIN (  SELECT FK\_Item\_Item\_Number, FK\_User\_Email from BoardGame  UNION  SELECT FK\_Item\_Item\_Number, FK\_User\_Email from PlayingCardGame  UNION  SELECT FK\_Item\_Item\_Number, FK\_User\_Email from CollectibleCardGame  UNION  SELECT FK\_Item\_Item\_Number, FK\_User\_Email from ComputerGame  UNION  SELECT FK\_Item\_Item\_Number, FK\_User\_Email from VideoGame  ) AS CI ON T.FK\_Proposer\_Item\_Item\_number=CI.FK\_Item\_Item\_Number  WHERE CI.FK\_User\_Email=’$UserID’ OR PI.FK\_User\_Email=’$UserID’  )  GROUP BY “My role” ; |

* + If Rejected % >=50%, highlight background in red
  + Below, display **Trade History Detail** form, provides detailed information related to trades, show Proposed Date, Accepted/Rejected Date, Trade Status, Response Time, My role, Proposed Item, Desired Item, Other User (Counterparty)

|  |
| --- |
| SELECT proposed\_data AS “Proposed Date”,  accept\_reject\_data AS “Accepted/Rejected Date”,  trade\_status AS “Trade status”,  ISNULL(proposed\_date-accept\_reject\_date, proposed\_date-FORMAT(  getdate(), MM/dd/yyyy) ) AS “Response time (days)”,  IF(PI.FK\_User\_Email=’$UserID’, “Proposer”, “Counterparty”) AS ‘My role’,  PI.title AS ‘Proposed Item’,  CI.title AS ‘Desired Item’,  IF( PI.FK\_User\_Email=’$UserID’, CU.nickname, PU.nickname ) AS ‘Other User’  FROM Trade AS T  INNER JOIN (  SELECT FK\_Item\_Item\_Number, FK\_User\_Email, title, from BoardGame  UNION  SELECT FK\_Item\_Item\_Number, FK\_User\_Email from PlayingCardGame  UNION  SELECT FK\_Item\_Item\_Number, FK\_User\_Email from CollectibleCardGame  UNION  SELECT FK\_Item\_Item\_Number, FK\_User\_Email from ComputerGame  UNION  SELECT FK\_Item\_Item\_Number, FK\_User\_Email from VideoGame  ) AS PI ON T.FK\_Proposer\_Item\_Item\_number=PI.FK\_Item\_Item\_Number  INNER JOIN (  SELECT FK\_Item\_Item\_Number, FK\_User\_Email from BoardGame  UNION  SELECT FK\_Item\_Item\_Number, FK\_User\_Email from PlayingCardGame  UNION  SELECT FK\_Item\_Item\_Number, FK\_User\_Email from CollectibleCardGame  UNION  SELECT FK\_Item\_Item\_Number, FK\_User\_Email from ComputerGame  UNION  SELECT FK\_Item\_Item\_Number, FK\_User\_Email from VideoGame  ) AS CI ON T.FK\_Proposer\_Item\_Item\_number=CI.FK\_Item\_Item\_Number  LEFT JOIN User AS PU ON PU.email=PI.FK\_User\_Email  LEFT JOIN User AS CU ON CU.email=CI.FK\_User\_Email  WHERE PI.FK\_User\_Email =’$UserID’ OR CI.FK\_User\_Email =’$UserID’  ORDER BY “Proposed Date” DESC, “Response time (days)” DESC; |

* After each row of **Trade History Detail** form, there is one ***Detail*** button, on click goes to **View Trade Detail** form
* A ***Back*** button should present, on click goes back to **Main Menu**

**View Trade Details** Chris G

Abstract Code

* Detail view breaks down into four sections
  + Trade Details
  + User Details
  + Proposed Item
  + Desired Item

|  |
| --- |
| DECLARE @Dlat DECIMAL(8,6)  DECLARE @Dlon DECIMAL(9,6)  DECLARE @a DECIMAL(9,6)  DECLARE @c DECIMAL(9,6)  DECLARE @R DECIMAL(6,2) = 3958.75  SELECT T.proposed\_date AS “Proposed Date”,  T.accept\_reject\_date AS “Accepted/Rejected Date”,  T.trade\_status AS “Trade status”,  IF(PI.FK\_User\_Email=’$UserID’, “Proposer”, “Counterparty”) AS ‘My role’,  ISNULL(T.proposed\_date- T.accept\_reject\_date, T.proposed\_date -FORMAT(  getdate(), MM/dd/yyyy) ) AS ‘Response time (days)’,  IF( PI.FK\_User\_Email=’$UserID’, CU.nickname, PU.nickname ) AS Nickname  @Dlat= Address.Latitude- Address.Latitude,  @Dlon= Address.Longitude- Address.Longtitude,  @a= SQUARE(SIN(@Dlat/2)) + COS(Address.Latitude)\* COS(Address.Latitude)\* SQUARE(SIN(@Dlon/2)),  @c= 2\*ATN2(SQRT(@a), SQRT(1-@a)),  @R \* @c AS Distance  IF( PI.FK\_User\_Email=’$UserID’, CU.first\_name + “ ” + CU.last\_name, PU.first\_name + “ ” + PU.last\_name ) AS Name  IF( PI.FK\_User\_Email=’$UserID’, CU.email, PU.email) AS Email  PI.FK\_Item\_Item\_Number AS ‘Item #’,  PI.title AS Title,  PI. game\_type AS "Game type”,  PI.condition AS Condition,  PI.Description AS Description,  CI.FK\_Item\_Item\_Number AS ‘Item #’,  CI.title AS Title,  CI. game\_type AS "Game type”,  CI.condition AS Condition,  FROM `Trade` AS T  INNER JOIN (  SELECT FK\_Item\_Item\_Number, FK\_User\_Email, title, “Board Game” AS game\_type, condition, description from BoardGame  UNION  SELECT FK\_Item\_Item\_Number, FK\_User\_Email, title, “Playing Card Game” AS game\_type, condition, description from PlayingCardGame  UNION  SELECT FK\_Item\_Item\_Number, FK\_User\_Email, title, “Collectible Card Game” AS game\_type, condition, description from CollectibleCardGame  UNION  SELECT FK\_Item\_Item\_Number, FK\_User\_Email, title, “Computer Game” AS game\_type, condition, description from ComputerGame  UNION  SELECT FK\_Item\_Item\_Number, FK\_User\_Email, title, “Video Game” AS game\_type, condition, description from VideoGame  ) AS PI ON T.FK\_Proposer\_Item\_Item\_number=PI.FK\_Item\_Item\_Number  INNER JOIN (  SELECT FK\_Item\_Item\_Number, FK\_User\_Email, title, “Board Game” AS game\_type, condition from BoardGame  UNION  SELECT FK\_Item\_Item\_Number, FK\_User\_Email, title, “Playing Card Game” AS game\_type, condition from PlayingCardGame  UNION  SELECT FK\_Item\_Item\_Number, FK\_User\_Email, title, “Collectible Card Game” AS game\_type, condition from CollectibleCardGame  UNION  SELECT FK\_Item\_Item\_Number, FK\_User\_Email, title, “Computer Game” AS game\_type, condition from ComputerGame  UNION  SELECT FK\_Item\_Item\_Number, FK\_User\_Email, title, “Video Game” AS game\_type, condition from VideoGame  ) AS CI ON T.FK\_Proposer\_Item\_Item\_number=CI.FK\_Item\_Item\_Number  LEFT JOIN User AS PU ON PU.email=PI.FK\_User\_Email  LEFT JOIN User AS CU ON CU.email=CI.FK\_User\_Email  INNER JOIN Address ON FK\_Address\_PostalCode=Address.postal\_code  WHERE PI.FK\_User\_Email =’$UserID’ OR CI.FK\_User\_Email =’$UserID’ |

* A ***Back*** button should present, on click goes back to **View Trade History**

**List Item**

Abstract Code

* Check if PendingTrades is greater than 2

|  |
| --- |
| SELECT COUNT(Item\_number) AS PendingTrades FROM (  SELECT item\_number FROM ((((BoardGame INNER JOIN PlayingCardGame) INNER JOIN CollectibleCardGame) INNER JOIN ComputerGame) INNER JOIN VideoGame)  WHERE user\_email = ’$UserID’  INTERSECT ALL  SELECT counter\_party\_item\_number AS 'item\_number' FROM trade ); |

* If user has > 2 unaccepted trades Then display error “Cannot list item. Check pending trades”
  + Back to ***Main Menu*** button
* Else:
  + Show **New Item Listing** form
  + Show empty title textbox
  + Show empty description textbox
  + Game type dropdown is populated with game type ENUM
  + Condition dropdown is populated with condition ENUM
  + If Collectable card game selected
    - Show text box for number of cards being offered
  + If video game selected
    - Show dropdown for platform

|  |
| --- |
| SELECT name AS 'PlatformName' from platform |

* + - Show dropdown for media – populated with media ENUM
  + If computer game selected
    - Show dropdown for platform – populated with computer game platform ENUM
* On click List Item button
* Validate user input
  + If failed validation, no query
  + Else:

|  |
| --- |
| game\_title = gt\_txtbox.GetValue()  game\_description = gd\_txtbox.GetValue()  game\_condition = gc\_combobox.GetSelectionIndex()  INSERT INTO Item () VALUES ();  INSERT INTO BoardGame (item\_number, title, description, game\_condition, owner\_email)  VALUES (SELECT MAX(item\_number), str(game\_title), str(game\_description), game\_condition, ’$UserID’); |

* Display screen with a success message pop up.
* End

### Search

**Abstract code**

* Begin
* Show a rundown of radio buttons which permit to look by watchword, in clients postal code, inside X miles, in determined postal code
* Client selects one of the options ($search\_option)
* On Click Search!:
  + Query the database to retrieve items which could be part of the new trade. These items should satisfy below three conditions:
    - Item should not belong to the user
    - Items should not be a part of an existing trade
    - With calculated distance based on postal\_code of $SessionID  
      (logged in user’s email) and the postal\_code of users items which satisfies one of the search conditions

|  |
| --- |
| WITH items\_union AS(  select  temp.item\_number,  title,  description,  game\_condition,  game\_type,  email  from  Item it  INNER JOIN (  (  select  item\_number,  title,  description,  game\_condition,  "Video game" as game\_type,  email  from  VideoGame  )  UNION  (  select  item\_number,  title,  description,  game\_condition,  "Computer game" as game\_type,  email  from  ComputerGame  )  UNION  (  select  item\_number,  title,  description,  game\_condition,  "Collectible card game" as game\_type,  email  from  CollectibleCardGame  )  UNION  (  select  item\_number,  title,  description,  game\_condition,  "Playing card game" as game\_type,  email  from  PlayingCardGame  )  UNION  (  select  item\_number,  title,  description,  game\_condition,  "Board game" as game\_type,  email  from  BoardGame  )  ) temp ON temp.item\_number = it.item\_number  ),  query\_on\_user AS(  *-- Used to get logged in TradePlazaUser's email*  SELECT  u.\*  FROM  TradePlazaUser u  WHERE  u.email = '$SessionID'  ),  accepted\_trade\_items AS (  *-- Used to get list of accepted items (proposer and counterparty) so that we can exclude them*  SELECT  proposed\_item\_number,  counter\_party\_item\_number,  trade\_status  FROM  Trade s  ),  items\_to\_find\_dist AS(  SELECT  item\_number,  game\_type,  title,  game\_condition,  description,  u.postal\_code AS item\_postal\_code,  u.email  FROM  items\_union i  LEFT JOIN TradePlazaUser u ON i.email = u.email  WHERE  u.email <> '$SessionID'  AND item\_number NOT IN(  (  SELECT  proposed\_item\_number  FROM  accepted\_trade\_items  )  UNION  (  SELECT  counter\_party\_item\_number  FROM  accepted\_trade\_items  )  )  ),  response\_time AS(  select  i.email,  ROUND(  avg(TIMESTAMPDIFF(DAY, accept\_reject\_date, NOW()), 1)  ) as Response\_Time  from  items\_union i  INNER JOIN TradePlazaUser u ON i.email = u.email  INNER JOIN Trade tr on i.item\_number = tr.counter\_party\_item\_number  where  trade\_status = "Accepted"  or trade\_status = "Rejected"  Group By  i.email  ),  user\_rank AS(  select  i.email,  Count(\*) as trade\_count,  CASE  WHEN trade\_count >= 10 THEN 'Alexandinium'  WHEN trade\_count >= 8  AND trade\_count <= 9 THEN 'Platinum'  WHEN trade\_count >= 6  AND trade\_count <= 7 THEN 'Gold'  WHEN trade\_count >= 4  AND trade\_count <= 5 THEN 'Silver'  WHEN trade\_count >= 3  AND trade\_count <= 4 THEN 'Bronze'  WHEN trade\_count >= 1  AND trade\_count <= 2 THEN 'Aluminium'  END as user\_rank  from  items\_union i  INNER JOIN TradePlazaUser u ON i.email = u.email  INNER JOIN ((  SELECT  proposed\_item\_number as tr\_item\_no  FROM  accepted\_trade\_items  where  trade\_status = "Accepted"  )  UNION  (  SELECT  counter\_party\_item\_number as tr\_item\_no  FROM  accepted\_trade\_items  where  trade\_status = "Accepted"  )) tr on i.item\_number = tr.tr\_item\_no  Group By  i.email  ),  lat\_lon AS(  SELECT  items\_to\_find\_dist.\*,  query\_on\_user.postal\_code AS user\_postal\_code,  RADIANS(a.latitude) AS lat1,  RADIANS(a2.latitude) AS lat2,  RADIANS(a.longitude) AS lon1,  RADIANS(a2.longitude) AS lon2,  RADIANS(a2.latitude - a.latitude) :: NUMERIC(9, 6) AS delta\_lat,  RADIANS(a2.longitude - a.longitude) :: NUMERIC(9, 6) AS delta\_lon  FROM  items\_to\_find\_dist  LEFT JOIN response\_time on items\_to\_find\_dist.email = response\_time.email  LEFT JOIN user\_rank on items\_to\_find\_dist.email = user\_rank.email  CROSS JOIN query\_on\_user  LEFT JOIN Address a ON items\_to\_find\_dist.item\_postal\_code = a.postal\_code  LEFT JOIN Address a2 ON query\_on\_user.postal\_code = a2.postal\_code  ),  haversine AS(  SELECT  DISTINCT ON (item\_postal\_code, user\_postal\_code) item\_postal\_code,  user\_postal\_code,  (SIN(delta\_lat / 2.0) \* SIN(delta\_lat / 2.0)) + (  COS(lat1) \* COS(lat2) \* SIN(delta\_lon / 2.0) \* SIN(delta\_lon / 2.0)  ) AS haversine\_a  FROM  lat\_lon  )  SELECT  la.item\_number,  la.game\_type,  la.title,  la.game\_condition,  la.description,  h.item\_postal\_code,  h.user\_postal\_code,  3958.75 \* 2 \* (ATAN2(SQRT(haversine\_a), SQRT(1 - haversine\_a))) AS distance  FROM  lat\_lon la  LEFT JOIN haversine h ON (  la.user\_postal\_code = h.user\_postal\_code  AND la.item\_postal\_code = h.item\_postal\_code  ) *--WHERE CLAUSE ADDED HERE* |

* If ($search\_option) = “search by keyword”
  + search only on name\_title and description for keyword
  + Add in to last line of above query where it is specified –WHERE

CLAUSES ADDED HERE–

WHERE

(

la.title LIKE '%$search\_keyword%'

OR la.description LIKE '%$search\_keyword%'

)

* If ($search\_option) = “search by in user’s postal code”

WHERE

h.item\_postal\_code = h.user\_postal\_code

* If ($search\_option) = “within X miles”
  + Take user input of X: ($miles)

WHERE

distance <= ‘ $ miles ’

* If ($search\_option) = “search by specified postal code”
  + Take user input of postal code: ($postal\_code\_search)

WHERE

h.item\_postal\_code = '$postal\_code\_search'

* Assuming that results are found:
* Show thing number, game sort, thing name/title, the condition and the initial 100 characters of the depiction (in the event that the portrayal is more noteworthy than 100 characters, place an ellipsis (… ) toward the finish to demonstrate it has been shortened) , normal reaction time and dealer position of other client
* In the event that hunt by catchphrase chose:
* Then, at that point, feature the fields that match the catchphrase in blue
* Assuming that pursuit by postcode is chosen AND the postcode is invalid:
* Then show a mistake message
* On click Detail:
* Divert to a matching View Item page
* ELSE
* Show a message "Sorry, no outcomes found!"
* END

**View Item**

**Abstract code**    
● Click through from Detail CTA In Search or Detail CTA in My Items  
   
● Determine if click through is from Search or My Items. Get item\_number  
 ($item\_number) based on the item that the user clicked

* If from My Items, display
  + item\_number, title, game type, condition, description (if not null)
  + If item listed is a “Video Game”, “Computer Game” or “Collectible Card Game”, display specific fields

|  |
| --- |
| WITH items\_union AS(  select  temp.item\_number,  title,  description,  game\_condition,  game\_type,  email  from  Item it  INNER JOIN (  (  select  item\_number,  title,  description,  game\_condition,  "Video game" as game\_type,  email,  null as number\_of\_cards,  null as cg\_platform,  media,  name as vg\_platform  from  VideoGame  INNER JOIN platform on VideoGame.platform\_id = platform.platform\_id  )  UNION  (  select  item\_number,  title,  description,  game\_condition,  "Computer game" as game\_type,  email,  null as number\_of\_cards,  platform as cg\_platform,  null as media,  null as vg\_platform  from  ComputerGame  )  UNION  (  select  item\_number,  title,  description,  game\_condition,  "Collectible card game" as game\_type,  email,  number\_of\_cards,  null as cg\_platform,  null as media,  null as vg\_platform  from  CollectibleCardGame  )  UNION  (  select  item\_number,  title,  description,  game\_condition,  "Playing card game" as game\_type,  email,  null as number\_of\_cards,  null as cg\_platform,  null as media,  null as vg\_platform  from  PlayingCardGame  )  UNION  (  select  item\_number,  title,  description,  game\_condition,  "Board game" as game\_type,  email,  null as number\_of\_cards,  null as cg\_platform,  null as media,  null as vg\_platform  from  BoardGame  )  ) temp ON temp.item\_number = it.item\_number  ),  SELECT  i.item\_number,  i.title,  i.game\_condition,  i.description,  i.game\_type,  i.vg\_platform,  i.media,  i.cg\_platform,  i.number\_of\_cards  FROM items\_union  WHERE  i.item\_number = '$item\_number'  AND i.email = '$SessionID' |

* If from Search, display the same information as listed in My Items, but with the addition of the following information about the other user (referred as counterparty).
  + nickname, city, state, postal code. Distance, item\_number, rank, response time can be taken from Search task and forwarded to this display
  + Using item\_number ($item\_number), perform query to get nickname of

counterparty with item\_number

SELECT

u.nickname

FROM

items\_union i

LEFT JOIN TradePlazaUser u ON i.email = u.email

WHERE

item\_number = ‘ $ item\_number ’

* Using postal code for item for counterparty (forwarded from Search), query city and state

SELECT

city,

state

FROM

Address

WHERE

postal\_code = '$postal\_code'

* If the user's postal code!= the item's owner postal code,
* Then display distance
* IF 0.0 < distance < 25.0 miles
* Then add green background
* IF 25.00 < distance < 50.0 miles
* Then add yellow background
* IF 50.00 < distance < 100.0 miles
* Then add orange background
* IF distance > 100.0 miles
* Then add red background
* OTHERWISE Hide distance
* Depending on the context, different elements will be displayed.
* If the user has not accepted < 2 trades, then display Propose Trade.
* OTHERWISE, Propose Trade is not displayed.
* When you click Propose Trade, you will be redirected to Add Trade Proposal

**List Item**

Abstract Code

* Check if PendingTrades is greater than 2

|  |
| --- |
| SELECT COUNT(Item\_number) AS PendingTrades FROM (  SELECT item\_number FROM ((((BoardGame INNER JOIN PlayingCardGame) INNER JOIN CollectibleCardGame) INNER JOIN ComputerGame) INNER JOIN VideoGame)  WHERE user\_email = ’$UserID’  INTERSECT ALL  SELECT counter\_party\_item\_number AS 'item\_number' FROM trade ); |

* If user has > 2 unaccepted trades Then display error “Cannot list item. Check pending trades”
  + Back to ***Main Menu*** button
* Else:
  + Show **New Item Listing** form
  + Show empty title textbox
  + Show empty description textbox
  + Game type dropdown is populated with game type ENUM
  + Condition dropdown is populated with condition ENUM
  + If Collectable card game selected
    - Show text box for number of cards being offered
  + If video game selected
    - Show dropdown for platform

|  |
| --- |
| SELECT name AS 'PlatformName' from platform |

* + - Show dropdown for media – populated with media ENUM
  + If computer game selected
    - Show dropdown for platform – populated with computer game platform ENUM
* On click List Item button
* Validate user input
  + If failed validation, no query
  + Else:

|  |
| --- |
| game\_title = gt\_txtbox.GetValue()  game\_description = gd\_txtbox.GetValue()  game\_condition = gc\_combobox.GetSelectionIndex()  INSERT INTO Item () VALUES ();  INSERT INTO BoardGame (item\_number, title, description, game\_condition, owner\_email)  VALUES (SELECT MAX(item\_number), str(game\_title), str(game\_description), game\_condition, ’$UserID’); |

* Display screen with a success message pop up.
* End

**My Items**

Abstract Code

* Get game counts for logged user

|  |
| --- |
| SELECT COUNT(item\_number) AS GameCount FROM BoardGame WHERE user\_email = ’$UserID’;  SELECT COUNT(item\_number) AS GameCount FROM CollectibleCardGame WHERE user\_email = ’$UserID’;  SELECT COUNT(item\_number) AS GameCount FROM ComputerGame WHERE user\_email = ’$UserID’;  SELECT COUNT(item\_number) AS GameCount FROM PlayingCardGame WHERE user\_email = ’$UserID’;  SELECT COUNT(item\_number) AS GameCount FROM VideoGame WHERE user\_email = ’$UserID’; |

* Get each item’s parameters and display them

|  |
| --- |
| SELECT title, condition, description from BoardGame WHERE user\_email = ’$UserID’;   * display all the user's listed BoardGame items. if len(description) > 100: description = description[:100] + "..."   SELECT title, condition, description from PlayingCardGame WHERE user\_email = ’$UserID’;   * display all the user's listed PlayingCardGame items. if len(description) > 100: description = description[:100] + "..."   SELECT title, condition, description from ComputerGame WHERE user\_email = ’$UserID’;   * display all the user's listed ComputerGame items. if len(description) > 100: description = description[:100] + "..."   SELECT title, condition, description from CollectibleCardGame WHERE user\_email = ’$UserID’;   * display all the user's listed CollectibleCardGame items. if len(description) > 100: description = description[:100] + "..."   SELECT title, condition, description from VideoGame WHERE user\_email = ’$UserID’;   * display all the user's listed VideoGame items. if len(description) > 100: description = description[:100] + "..." |

* For each item, display a button to bring up a detailed item view