**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