Group 7
Project name: Gift Registry System
Sprint 3

HIGH LEVEL REQUIREMENTS

Initial user roles

User Role	**Description**
Inviter	Inviters who have a registered account with the system. These Inviters can create registry and register their event and add items to their registry. Also they can create invitee list and send a message. They can view the registry list who is buying the items for Them. And view the wishlist product items which are ordered. Also they can track the order items. Inviters can review the ordered items.
ProductManager	ProductManager will add/remove the product information. They add or delete product wishlist category and check the availability of the product in inventory.
Invitee	Invitee who have free access to inviter's registry. They can buy the registry items. Also they add shopping cart and place an order. They -can view their order history and delivery status.

Initial user story descriptions

Story ID	Story description
US14	As an Invitee, I want to place an order from my shopping cart.
US16	As an Inviter, I want to view wishlist list so that I can check what items are ordered by which invitee.
US17	As an Invitee, I want to check my order history and track the delivery status.
US18	As an Invitee, I want to cancel order before shipment.

CONCEPTUAL DESIGN

Entity: **ProductManager**
Attributes:
- username
- name[composite]
  - first_name
  - middle_name
  - last_name
- password
- address [composite]
  - address_line1
  - address_line2
  - city
  - state
  - zip_code
- email_address
- phone_number

Entity: **Product**
Attributes:
- id
- name
- description

- unit_price
- quantity

Entity: **Inviter**

Attributes:

- Username
- password
- name [composite]
  - first_name
  - middle_name
  - last_name
- phone_number
- address[composite]
  - address_line_1
  - address_line_2
  - city
  - state
  - Zip_code
- email_address

Entity: **Gathering**

Attributes:

- id
- name
- date
- time
- description
- venue[composite]
  - address_line_1
  - address_line_2
  - city
  - state
  - zip_code

Entity: **Invitee**
Attributes:
email_address
password
name [composite]
first_name
middle_name
last_name
phone_number
address[composite]
address_line_1
address_line_2
city
state
Zip_code

Entity: **Order**
Attributes:
Order_number
quantity
unit_price
total_price
order_date
card_number
cancel_yn

Entity: **Delivery**
Attributes:
id
tracking_number
deliver_method
due_date
status

Relationship: **ProductManager** adds **Product**
Cardinality: One to Many
Participation:

ProductManager has partial participation
Product has total participation

Relationship: **ProductManager** adds **company**
Cardinality:One to one
Participation:

ProductManager has total participation
Company has total participation

Relationship:**ProductManager** adds **category**
Cardinality**:**One to many
Participation:

Productmanager has partial participation
Category has total participation

Relationship:**Product** has **Company**
Cardinality:many to one
Participation:

Product has total participation
Category has total participation

Relationship:**Product** has **Category**
Cardinality:one to one
Participation:

Product has partial participation
Category has total participation

Relationship: **Inviter** creates **Gathering**
Cardinality: One to Many
Participation:

Inviter has partial participation
Gathering has total participation

Inviter adds Guests to the gathering

Relationship: **Inviter** adds **Guest**
Cardinality:Many to many
Participation:

Inviter has partial participation
Guest has Total participation

Relationship:**Gathering** has **Guest**
Cardinality:Many to many
Participation:

Gathering has partial participation
Guest has total participation

Inviter adds products to the wishlist

Relationship:**Inviter** adds **product**
Cardinality:One to many
Participation:

Inviter has partial participation
Product has total participation

Relationship:**Gathering** has **WishlistProduct**
Cardinality:One to many
Participation:

Both will have total participation

Invitee adds products to the cart

Relationship:**Invitee** adds **product**
Cardinality:Many to many
Participation:

Invitee will have partial participation
Product will have total participation

Relationship:**Cart** has **product**
Cardinality:Many to many

Cart will have partial participation
Product will have total participation

Relationship:**Invitee** RSVP’s to **Gathering**
Cardinality:Many to many
Invitee has partial participation
Gathering has partial participation

Relationship: **order** has **product**
Cardinality: many to many
Participation:
Order has partial participation
Product has total participation

Invitee check delivery of order

Relationship: **Invitee** check **delivery** of **order**
Cardinality: one to many
Participation:
Invitee has partial participation
delivery has total participation

Relationship: **order has a Delivery**
Cardinality: one to many
Participation:
delivery has total participation
Order has partial participation

Relationship: **Inviter** return **product**
Cardinality: one to many
Participation:
Inviter has partial participation
Product has total participation

LOGICAL DESIGN

Table: **ProductManager**

Columns:

Username
password
first_name
middle_name
last_name
address_line1
address_line2
city
state
zipcode
email_address
phone_number

Primary key Justification: username will be unique for each Product Manager while signing up. So username becomes the primary key of the table ProductManager.

Highest normalization level: <3NF>

Justification : Generally address_line1, city, and state have a functional dependency on zipcode, but we are considering our application worldwide and there can be the areas who share the same zip code. Hence, we the zipcode table has not been separated.

Also, 4NF is inefficient for this table because there is a concern about performance degradation when it comes to joining the tables and there are generally not many changes to these entities.

Index 1: < (non-clustered)>

Columns: first_name

Justification: First_name is the very likely to come up for search queries when trying to get information about a specific productManager.

Table: **ProductCompany**

Columns:

id

name

Primary key Justification: id will be unique for each ProductCompany. So id becomes the primary key of the table .

Highest normalization level: <4NF>

Indexes:

Index 1: < (clustered)>

Columns: id

Justification: In many cases, the productCompany table join another table with id

Table: **ProductCategory**

Columns:

id

name

Primary key Justification: id will be unique for each ProductCategory. Hence, it becomes the primary key for the table **ProductCategory**.

Highest normalization level: <4NF>

Indexes:

Index 1: < (clustered)>

Columns: id

Justification: In many cases, the productCategory table join another table with id

Table: **Product**

Columns:

id
name
description
unit_price
quantity
company_id[foreign key;references id of ProductCompany]
pm_username[foreign key;references username of ProductManager]
category_id [foreign key;references id of ProductCategory]

Foreign key approach with the column pm_username.

Primary key Justification: id will be unique for each Product. Hence, it becomes the primary key for the table Product.

Foreign key justification: As username is the primary key of the table ProductManager, it can perfectly connect ProductManager table with Product table to keep a track which Product managers are adding the which products.

Foreign key justification: As id is the primary key of the table ProductCompany, it can perfectly connect ProductCompany table with the Product table to identify the company of the particular product.

Foreign key justification:As id is the primary key of the table Product*Category , it can perfectly connect* Product*Category table with the Product table to identify the category of a particular product.*

Highest normalization level: <3NF>
Justification :
4NF is inefficient for this table because there is a concern about performance degradation when it comes to joining the tables and there are generally not many changes to these entities.

Indexes:

Index 1: <(clustered)>
Columns: id
Justification: In many cases, the product table join another table with id

Table: **Inviter**

Columns:

username
password
first_name
middle_name
last_name
email_address
phone_number
address_line1
address_line2
city
state
zipcode

Primary key Justification: username will be unique for each Inviter. Hence, it becomes the primary key for the table Inviter.

Highest normalization level: <3NF>
Justification :
Justification : Generally address_line1, city, and state have a functional dependency on zipcode, but we are considering our application worldwide and there can be the areas who share the same zip code. Hence, we the zipcode table has not been separated.

4NF is inefficient for this table because there is a concern about performance degradation due to joining when performing high normalization while not having many changes to this entity.

Indexes:
Index 1: <type (clustered)>
Columns: username
Justification: In many cases, the Inviter table join another table with username

Table: **Gathering**

Columns:

Id
name
date
description
address_line1
address_line2
city
state
zipcode
inviter_username[foreign key;references username of **Inviter**]

Primary key Justification: id will be unique for each Gathering. Hence, it becomes the primary key for the table Gathering.

Foreign key justification: As *username* is the primary key of the table *Inviter*, it can perfectly connect Gathering table with *Inviter* table to identify which Inviter has created the gathering. Hence, inviter_username becomes the foreign key for the table Gathering.

Highest normalization level: <3NF>

Justification :

Justification : Generally address_line1, city, and state have a functional dependency on zipcode, but we are considering our application worldwide and there can be the areas who share the same zip code. Hence, we the zipcode table has not been separated.

4NF is inefficient for this table because there is a concern about performance degradation due to joining when performing high normalization with not many changes to the table.

Indexes:

Index 1: <non-clustered)>

Columns: name

Justification: In many cases, the product table is queried by name and ordering by name

Table: **GatheringGuests**

Columns:

email_address[Foreign key Primary key]

gathering_id[Foreign key;references id of Gathering]

Cross Reference approach because not all guests are signed up as invitees.

Primary key Justification: email will be unique for each person. Hence, it becomes the primary key for the table Guests.

*Foreign key justification:*As every gathering has it's own guests associating gathering with it's id is the best way to connect guests to a particular gathering.

Highest normalization level: <4NF>

Indexes:

Index 1: <type (clustered)>

Columns: email_address

Justification: In many cases, the GatheringGuest table join another table with email_address

Table: **Invitee**

Columns:

email_address

password

first_name

middle_name

last_name

phone_number

address_line1

address_line2

city

state
zipcode

Primary key Justification: email_address will be unique for each Invitee and they will be added to the guests for a gathering using their email address making it easy to associate rather than having a username. Hence, it becomes the primary key for the table **Invitee**.

Highest normalization level: <3NF>
Justification :
Justification : Generally address_line1, city, and state have a functional dependency on zipcode, but we are considering our application worldwide and there can be the areas who share the same zip code. Hence, we the zipcode table has not been separated.

4NF is inefficient for this table because there is a concern about performance degradation due to joining when performing high normalization with not many changes to the entity.

Indexes:
Index 1: <type (clustered)>
Columns: email_address
Justification: In many cases, the Invitee table join another table with email_address

Table: **InviteStatus**
Columns:
id
RSVP
gathering_id[foreign key;references id of **Gathering**]
invitee_email[foreign key;references email_address of **Invitee**]
Cross Reference approach since one Invitee may be invited to multiple gathering while one gathering can have multiple invitee's with their response stored in InviteStatus making this a viable option.

Foreign key justification: As *id* is the primary key of the table *Gathering*, it can perfectly connect Gathering table with *InviteStatus* table to identify whether *Invitee* has RSVP'd to a particular gathering. Hence, gathering_id becomes the foreign key for the table **Invitee**.

Foreign key justification: As *invitee_email* is the primary key of the table *Invitee*, it can perfectly connect Invitee entity with *InviteStatus* table to identify which Invitee has RSVP'd to which gathering . Hence, inviter_username becomes the foreign key for the table **InviteStatus**.

Highest normalization level: <3NF>
Justification :
4NF is inefficient for this table because there is a concern about performance degradation due to joining when performing high normalization.

Indexes:
Index 1: <type (clustered)>
Columns: id
Justification: In many cases, the InviteStatus table join another table with id

Table: **WishlistProduct**
Columns:
id
Quantity
gathering_id[foreign key;references id of **Gathering**]
product_id[foreign key;references id of **Product**]

Primary key Justification: id will be unique for each **WishlistProduct**. Hence, it becomes the primary key for the table **WishlistProduct**.

Foreign key justification: As *id* is the primary key of the table *Gathering*, it can perfectly connect *Gathering* table with *WishlistProduct* table to identify which *Gathering* has the particular wishlistProduct. Hence, gathering_id becomes the foreign key for the table **WishlistProduct**.

Foreign key justification: As *id* is the primary key of the table *Product*, it can perfectly connect *WishlistProduct* table with *Product* table to identify which products are in there in the inventory. Hence, product_id becomes the foreign key for the table **WishlistProduct**.

Highest normalization level: <3NF>
Justification :
4NF is inefficient for this table because there is a concern about performance degradation due to joining when performing high normalization.

Indexes:
Index 1: <type (clustered)>
Columns: id
Justification: In many cases, the WishlistProduct table join another table with id

Table: **Cart**

Columns:

id
invitee_email[foreign key;references email_address of **Invitee**]
gathering_id[foreign key;references id of **Gathering**]

Primary key Justification: id will be unique for each ***Cart****. Hence, it becomes the primary key for the table* **Cart***.*

Foreign key justification: As email_address is the primary key of the invitee table ,it can perfectly connect Cart table with Invitee table to identify the invitee associated with that particular cart.
Foreign key justification: As id is the primary key of the gathering table ,it can perfectly connect Cart table with gathering table to identify the invitee associated with that particular cart.

Highest normalization level: <3NF>
Justification :
4NF is inefficient for this table because there is a concern about performance degradation due to joining when performing high normalization.
Indexes:
Index 1: <type (clustered)>

Columns: id

Justification: In many cases, the cart table join another table with id

*Table:***CartProduct**

Columns:

id

cart_id[foreign key;references id of **Cart**]

product_id[foreign key;references id of **Product**]

quantity

Primary key Justification: id will be unique for each ***CartProduct*** *autogenerated.*

Foreign key justification: One cart can have many products linking carts_id to product_id would connect tables cart and product respectively.

Highest normalization level: <3NF>

Justification :

4NF is inefficient for this table because there is a concern about performance degradation due to joining when performing high normalization.

Indexes:

Index 1: <type (clustered)>

Columns: id

Justification: In many cases, the cartProduct table join another table with id

*Table:***Orders**

Columns:

Id

Invitee_email[foreign key;references email_address of **invitee**]

gathering_id[foreign key;references id of **gathering**]

Order_date

card_number[foreign key;references card_number of **Card**]

Primary key Justification: id will be unique for each ***Orders*** *and is autogenerated.*

Foreign key justification: An order is connected with a particular invitee and hence invitee table's email_address works as the foreign key in order table as invitee_email.

An order is also connected with a particular gathering and hence gathering table's id works as the foreign key in order table as gathering_id.

An order can be placed by one card so card_number in card table the foreign key in order table.

Highest normalization level: <3NF>
Justification :
4NF is inefficient for this table because there is a concern about performance degradation due to joining when performing high normalization.

Indexes:
Index 1: <non-clustered)>
Columns: invitee_email
Justification: In many cases, the Orders table is queried by invitee_email and ordering by invitee_email

*Table:**Card***

Columns:

card_number
Expdate
type

Primary key Justification: card_number will be unique for each card so it can be considered as the primary key of the table Card.

Highest normalization level: <3NF>
Justification :
4NF is inefficient for this table because there is a concern about performance degradation due to joining when performing high normalization.

Indexes:
Index 1: <type (clustered)>
Columns: card_number

Justification: In many cases, the card table join another table with card_number

Table: **OrderProduct**

Columns:

id

order_number[foreign key;references id of **Order**]

product_id[foreign key;references id of **Product**]

quantity

tracking_number[foreign key;references tracking_number of **Delivery**]

cancelYN

Cross Reference approach because not all cart items are ordered.

Primary key Justification: id will be unique for each order products so it works as the primary key.

Foreign key justification: As *id* is the primary key of the table *Product*, it can perfectly connect *OrderProduct* table with *Product* table to identify which products are in there in the inventory. Hence, product_id becomes the foreign key for the table

Foreign key justification: As *tracking_number i*s the primary key of the table *Delivery*, it can perfectly connect *OrderProduct* table with *Delivery* table to identify which products are in there in the inventory. Hence, tracking_number becomes the foreign key for the table

Highest normalization level: <3NF>

Justification :

4NF is inefficient for this table because there is a concern about performance degradation due to joining when performing high normalization.

Indexes:

Index 1: <type (clustered)>

Columns: id

Justification: In many cases, the orderProduct table join another table with id

Table: **Delivery**

Columns:

tracking_number

due_date

status

Cross Reference approach because not all cart items are ordered.

Primary key Justification: tracking_number will be unique for each order so it works as the primary key.

Highest normalization level: <3NF>

Justification :

4NF is inefficient for this table because there is a concern about performance degradation due to joining when performing high normalization.

Indexes:

Index 1: <type (clustered)>

Columns: tracking_number

Justification: In many cases, the Delivery table join another table with tracking_number