Final Project Report

LendKart

Submitted to
Yashvardhan Sharma
yash@pilani.bits-pilani.ac.in

Submitted by
Aastha Sanghi (2016H112159P)
Aparajita (2016H112165P)
Priti Sharma(2016H1121168P)
Sarita Sharma(2016H112163P)
Sushmitha M. (2016H112173P)

11/28/2016

Contents

List of tables

List of figures

1. INTRODUCTION

LendKart, a one of its own kind app, provides a service for lending and buying a variety of goods and services under the same roof.

Nowadays e-commerce provides an easy way to buy and sell products to a large customer base and at the same time, online classifieds provide a medium to post advertisements. However, there are enormous numbers of e-commerce and online classifieds just a click away, then why the user needs to use our application, LendKart?

With growing new markets, customer choice has widened and is posing a big challenge to ensure customer welfare. There are situations where the user prefers to rent or lend a product instead of buying it. So, we come up with a collaboration of E-commerce and online classified “LendKart, Apna Profit toh banta hai!”. LendKart is an provides a platform for the user to market his product as well as fetch customers for his product. It follows a C2C model (consumer to consumer). This application helps user to maximize the utility of his product on a beneficial note.

It has wide scope as it is neither location bounded or category specific. Any community, residential complexes, colleges, companies can use this app**.**

2. SYSTEM FUNCTIONS

System functions describe what a system is supposed to do thus system functions for LendKart are given below. The category field in tables defines the degree of visibility to the user. *Hidden* means that the function is not visible by any actors and *evident* indicates that the user or external interface can see the function actively or passively.

2.1 Basic Functions

Below functions are essential for better understandings of the system.

Ref #	Functions	Category
R1.1	Create post	Evident
R1.2	View lent items history	Evident
R1.3	View borrowed items history	Evident
R1.4	Approve borrower	Evident
R1.5	Finish settlement	Evident
R1.6	Rate user	Evident
R1.7	Filter posts by category	Evident
R1.8	View post	Evident
R1.9	Add post to wishlist	Evident
R1.10	Show interest over post	Evident

R1.11	Update user profile	Evident
R1.12	View wishlist	Evident

TABLE 2.1 SYSTEM FUNCTIONS

3. SYSTEM ATTRIBUTES

System attributes are non-functional qualities or characteristics of the system. Each system attribute was labeled as a *must* or *want*. Attributes labeled with *must* are mandatory attributes of the system, and attributes marked with *want* are desirable but not required.

Attribute	Details and Boundary Constraints	Category
Easy to use GUI	Android Platform	Must
Backend	Django1.10	Must
Database	PostgreSQL9.5	Must
Object Oriented Design	This allows the product to be easily extended in future development	Must

TABLE 3.1 SYSTEM ATTRIBUTES

4. USE CASES

Below user view of 'Lendkart' gives description of expanded use cases and the sequence of actor actions and system responses for each use case.

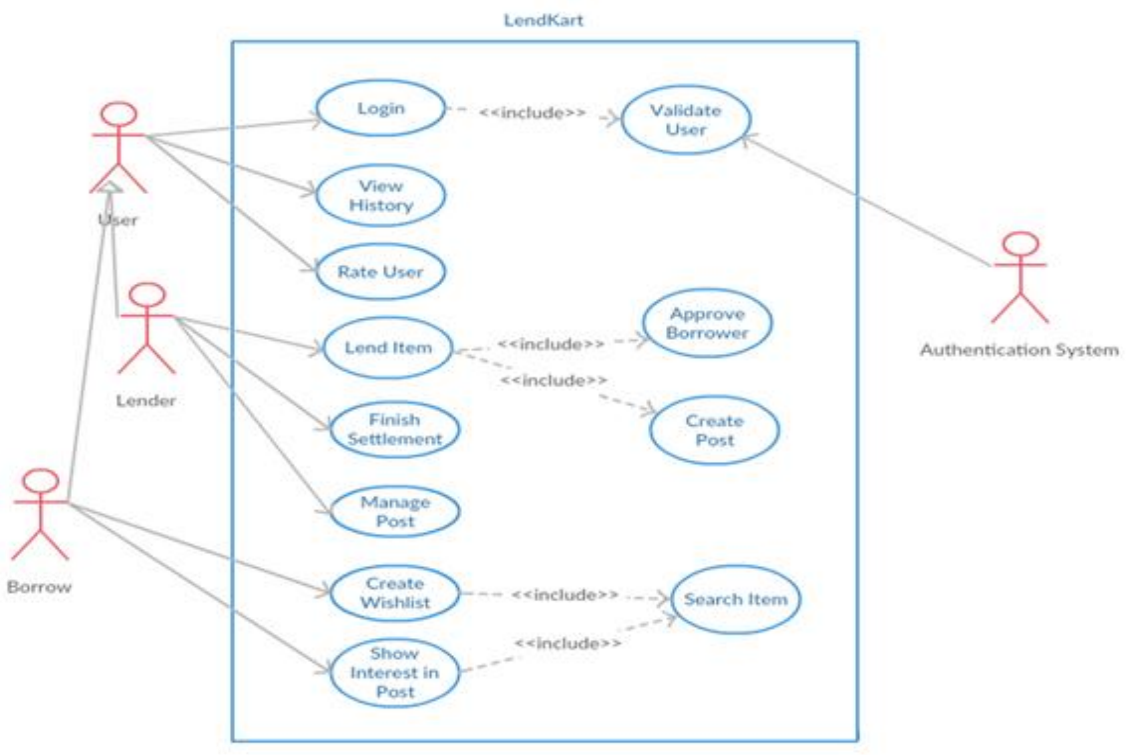


FIGURE 4.1 USE CASE DIAGRAM

4.1 Use Case Descriptions

Below is the description for major use cases shown in use case diagram

Name	R1.1 Create Post	
Brief Description	A user of the System creates a new post for the item he/she wants to lend	
Actor(s)	End user	
Flow of Events		
Main Success Scenario		
This use case starts when an end user is logged in to the system and goes to the lend page. 1. A form is displayed to the user asking for the information required to create a post. 2. The user fills in the item details and uploads photo of the item (optional) and clicks on 'Create Post' 3. System validates the information and if found correct displays the message "Post has been successfully created". 4. The newly created post details are shown to the user 5. The use case ends.		
Alternate Flows		
Title	**Description**	
Invalid Information Entered	1. User clicks submit after entering information system asked for. 2. System displays information with appropriate message to correct invalid information. 3. User re-enters information.	
Pre-Conditions		
User must be logged into the LendKart App		
Post-Conditions		
Title		**Description**
Success		The newly created post details are displayed

TABLE 4.1.1 CREATE POST DESCRIPTION

Name	R1.2 View lent items history
Brief Description	A user of the System views the list of items that he has lent to other users
Actor(s)	End user
Flow of Events	
Main Success Scenario	
This use case starts when an end user is logged in to the system. 1. User opens the side navigation menu. 2. He selects the 'View Lent History' option. 3. A list of items that has been lent by the user is shown. 4. The use case ends.	
Alternate Flows	
Title	**Description**

NA	NA
Pre-Conditions	
User must be logged into the LendKart App	
Post-Conditions	
Title	**Description**
Success	1. The list of items that has been lent by the user is shown.

TABLE 4.1.2 VIEW LENT HISTORY DESCRIPTION

Name	R1.3 View borrowed items history
Brief Description	A user of the System views the list of items that he has borrowed from other users
Actor(s)	End user
Flow of Events	
Main Success Scenario	
This use case starts when an end user is logged in to the system. 1. User opens the side navigation menu. 2. He selects the 'View Borrow History' option. 3. A list of items that has been borrowed by the user is shown. 4. The use case ends.	
Alternate Flows	
Title	**Description**
NA	NA
Pre-Conditions	
User must be logged into the LendKart App	
Post-Conditions	
Title	**Description**
Success	The list of items that has been borrowed by the user is shown.

TABLE 4.1.3 VIEW BORROWED HISTORY DESCRIPTION

Name	R1.4 Approve borrower
Brief Description	End user approves the request of an interested borrower for his post
Actor(s)	End user
Flow of Events	
Main Success Scenario	
This use case starts when an end user receives an interest for his post. 1. User views the list of interested borrowers for his post. 2. User reviews the borrowers' profiles. 3. The user approves the borrower's request. *4.* The use case ends.	

Alternate Flows	
Title	**Description**
User cancels the borrower’s request	Borrower’s request is cancelled and the borrower’s name is removed from the interested borrowers’ list.
Pre-Conditions	
End user receives an interest for his post	
Post-Conditions	
Title	**Description**
Success	The borrower’s request is approved and the borrower is notified of the same

TABLE 4.1.4 APPROVE BORROWER DESCRIPTION

Name	R1.5 Finish settlement
Brief Description	End user finishes the settlement once the borrower returns the borrowed items.
Actor(s)	End user
Flow of Events	
Main Success Scenario	
This use case starts when the borrower returns the borrowed items to the lender 1. User opens the post for the item for which the borrower has returned the items. 2. User clicks the “Settle Up” button. 3. The transaction is settled. *4.* The use case ends.	
Alternate Flows	
Title	**Description**
NA	NA
Pre-Conditions	
The borrower returns the borrowed items to the lender.	
Post-Conditions	
Title	**Description**
Success	The transaction is settled and the borrower is notified about the same

TABLE 4.1.5 FINISH SETTLEMENT DESCRIPTION

Name	R1.6 Rate user
Brief Description	Once a transaction is settled, both the lender and borrower rate each other.
Actor(s)	End user
Flow of Events	
Main Success Scenario	

This use case starts once the transaction is settled by the lender. 1. User clicks on "Rate User" 2. User rates the lender or borrower 3. User clicks on "Submit rating" *4.* The use case ends.	
Alternate Flows	
Title	**Description**
NA	NA
Pre-Conditions	
The transaction should be settled by the lender.	
Post-Conditions	
Title	**Description**
Success	The user's rating is stored in the backend.

TABLE 4.1.6 RATE USER DESCRIPTION

Name	R1.7 Update user's profile
Brief Description	The user updates his profile.
Actor(s)	End user
Flow of Events	
Main Success Scenario	
This use case starts when an end user is logged in to the system. 1. The user opens the side navigation menu. 2. User clicks on the "My Profile". 3. User enters information and clicks on "Save". 4. System validates the user's information. If the information is valid, then the system saves the information in the backend. *5.* The use case ends.	
Alternate Flows	
Title	**Description**
User enters invalid information	1. User clicks submit after entering information system asked for. 2. System displays information with appropriate message to correct invalid information. 3. User re-enters information.
Pre-Conditions	
The user must be logged into the LendKart app.	
Post-Conditions	
Title	**Description**
Success	The user's profile is updated in the backend.

TABLE 4.1.7 UPDATE USER PROFILE DESCRIPTION

Name	R1.8 View wishlist
Brief Description	A user of the System views the list of items that he has added to his wishlist
Actor(s)	End user

Flow of Events	
Main Success Scenario	
This use case starts when an end user is logged in to the system. 1. User opens the side navigation menu. 2. He selects the ‘View Wishlist’ option. 3. A list of items that has been added to user’s wishlist is displayed. *4.* The use case ends.	
Alternate Flows	
Title	**Description**
NA	NA
Pre-Conditions	
User must be logged into the LendKart App	
Post-Conditions	
Title	**Description**
Success	The list of items that has been added to user’s wishlist is shown.

TABLE 4.1.8 VIEW WISHLIST DESCRIPTION

5. CONCEPTUAL DIAGRAM

The conceptual model in is a model of significant concepts, attributes, and associations in the problem domain. The model also provides information about relationships and multiplicity in the interactions between concepts. Creating the conceptual model helps to gain a better perspective through an object-oriented analysis of the problem domain.

The description of the conceptual classes is as follows:

1. **User_Profile**: This class contains the information about users like username, email, contact number etc.
2. **Borrower**: This class contains the borrower details.
3. **Lender**: This class contains the lender details.
4. **Feedback**: This class describes the feedback details given by users to each other. Users can give feedback to each other once the transaction has been completed.
5. **Posts**: This class contains the information regarding the individual posts.
6. **Post_Settlement**: This class contains the information regarding the settlement of the transaction.
7. **Interested_Borrower**: This class contains the information regarding the list of borrowers who are interested in a particular post.
8. **WishList**: This class contains the information regarding the list of items that have been added by the user to his wishlist. These are the items that the user likes but for which user has not shown interest.
9. **Category**: This class describes the different categories of the post like electronics, books etc.

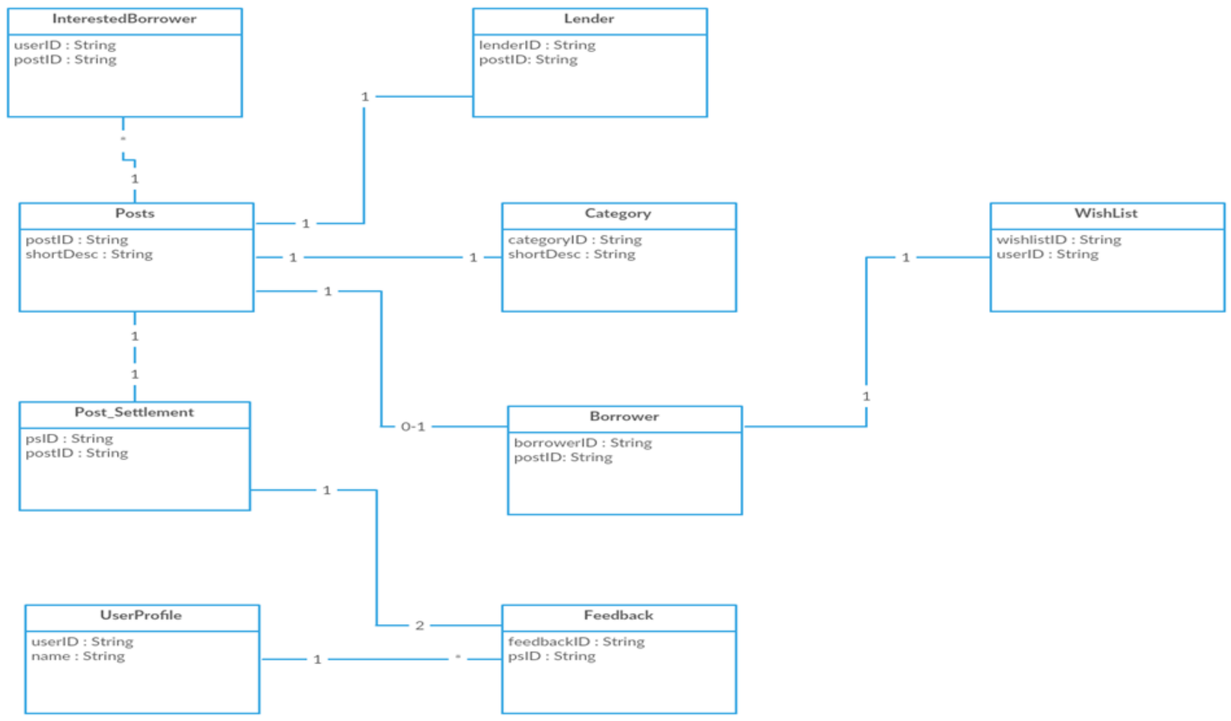


FIGURE 5. DOMAIN MODEL

7. SYSTEM BEHAVIOUR

Before moving on to logical design, System Behavior needs to be analyzed to define how a system behaves. System Behavior is a description of what a system does, while hiding the actual implementation. For this document, following diagrams are used to demonstrate the important system behaviors of 'LendKart'.

7.1 System Sequence Diagrams

- Lender creates post by submitting details post after which system returns the status if post is created successfully.
- Borrower search for list of posts for a item and system returns corresponding category of posts. Also he can show interest in item.
- Lender approves borrower request sent by borrower interested in post. Lender can finish settlement if he has successfully received the amount for lent item. Lender can submit the feedback of the borrower then.
- Lender and Borrower both view ratings.

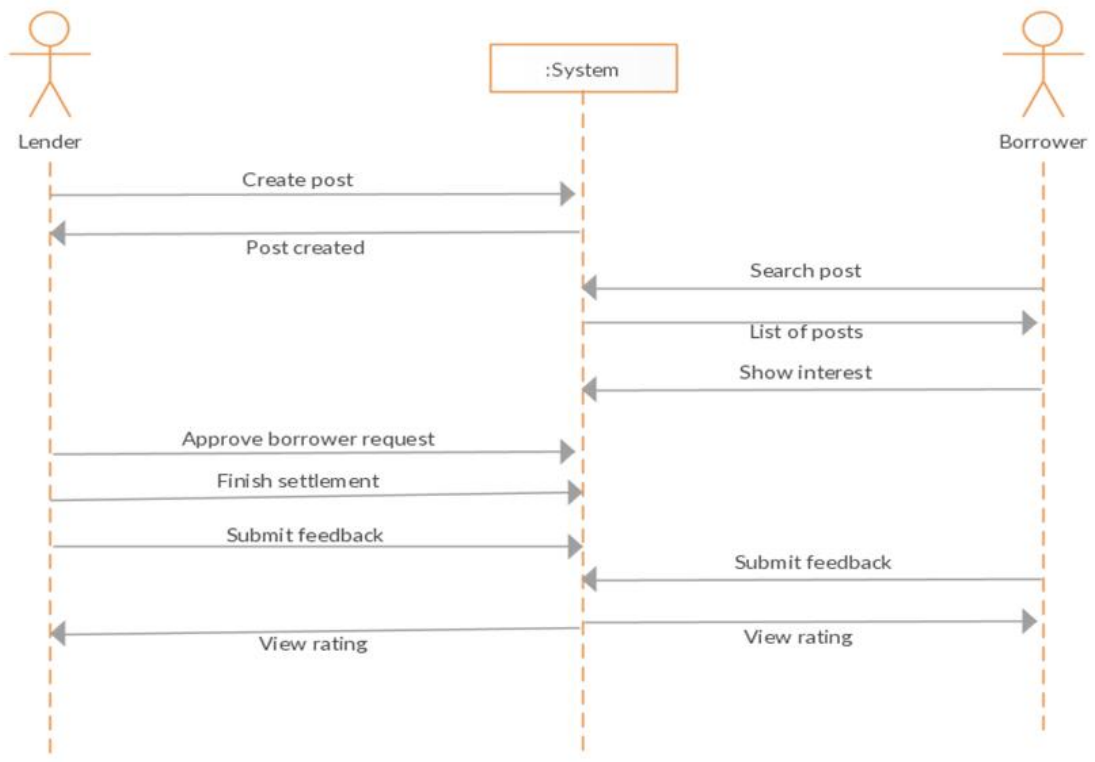


FIGURE 6 SYSTEM SEUQENCE DIAGRAM

7.2 System contracts

System Contracts describe detailed system behavior in terms of state changes of objects in the Domain Model, after a system operation has executed.

Contracts may be defined for system operations – operations that system as a black box offers in its public interface to handle incoming system events. System operations can be identified by discovering these system events.

The entire set of system operations, across all use cases, defines the public system interface, viewing the system as a single component or class.

7.2.1 Contract for create post

Name	CreatePost()
Use Case	R1.1 Create Post
Pre-Conditions	
User must be logged into the LendKart App	
Post-Conditions	
• Post, instance created. • Post object associated with a Category and Tenure	

7.2.2 Contract for approve borrower request

Name	ApproveBorowwerRequest()
Use Case	R1.4 Approve borrower
Pre-Conditions	
End user receives an interest for his post	
Post-Conditions	
• InterestedBorrower object created • IntrestedBoorower object associated with Post object • Noofinterestedborrower attribute change in Post.	

7.2.3 Contract for finish settlement

Name	FinishSettlement()
Use Case	R1.5 Finish settlement
Pre-Conditions	
The borrower returns the borrowed items to the lender.	
Post-Conditions	
• Post,InterestedBorrower state change.Borrower instance created. • assoc • isAvailaible in Post is set to no and isactive in Interestedborrower is set to no..	

7.2.4 Contract for lender submit feedback

Name	Submitfeedback()(From lender view)
Use Case	R1.5 Finish settlement
Pre-Conditions	
The borrower returns the borrowed items to the lender.	
Post-Conditions	
• Feedback object created.. • Feedback object associated with Lender and Post object.	

7.2.5 Contract for borrower submit feedback

Name	Submitfeedback()(Borrower view)
Use Case	R1.5 Finish settlement
Pre-Conditions	
The borrower returns the borrowed items to the lender.	
Post-Conditions	

- Feedback object created..
- Feedback object associated with Borrower and Post object.

8. INTERACTION DIAGRAMS

8.1 COLLABRATION DIAGRAM

Collaboration Diagrams illustrate the interactions between objects in a graph format. Following section detail some of the significant system behaviors with the help of collaboration diagrams.

8.1.1 Collaboration diagram to create post

Create post page receives item details like description, image, duration and post ID is stored in post catalogue corresponding to details received in create post page.

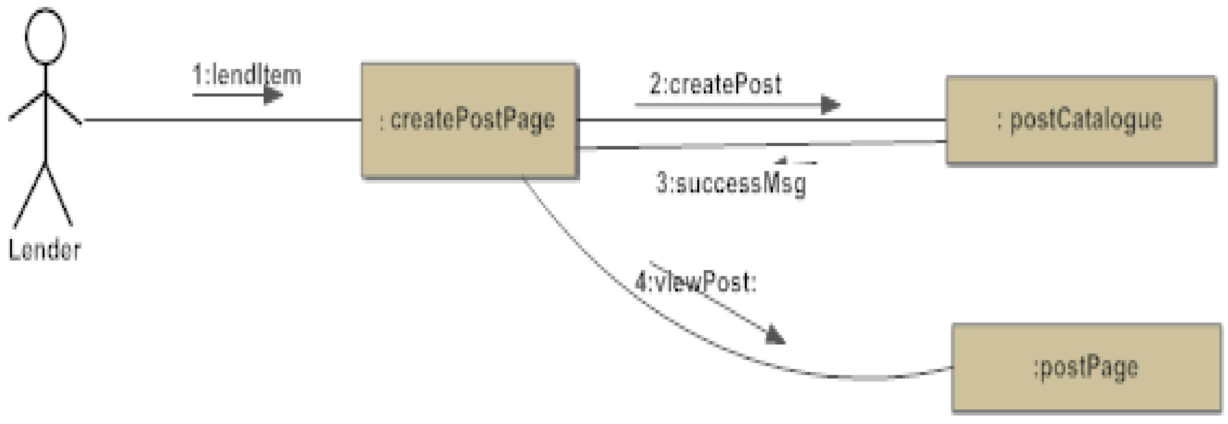


FIGURE 8.1.1 CREATE POST COLLABRATION DIAGRAM

8.1.2Collaboration diagram to borrow post

In borrow page on clicking a specific category control will flow to view PostPage with which user can select a particular Post .Borrower can show interest after which post will be passed to cart class.

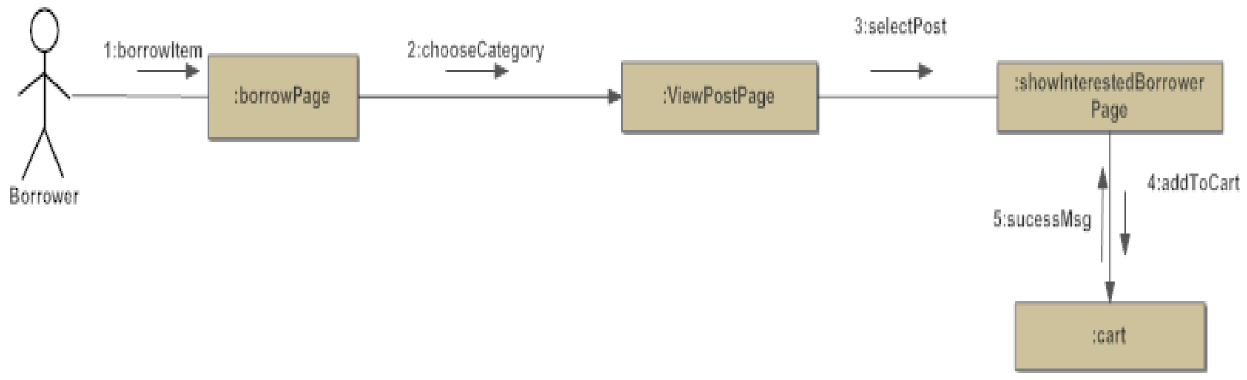


FIGURE 8.1.2 BORROW ITEM COLLABRATION DIAGRAM

8.1.3 Collaboration diagram to approve post

When lender triggers lentHistory and on lentHisntory user can view all the lent items. For a lent item choose any post and control will flow to postDetails page. Select interested borrower and control will flow to showInterestedBorrower page. A lender can approve request for one borrower .

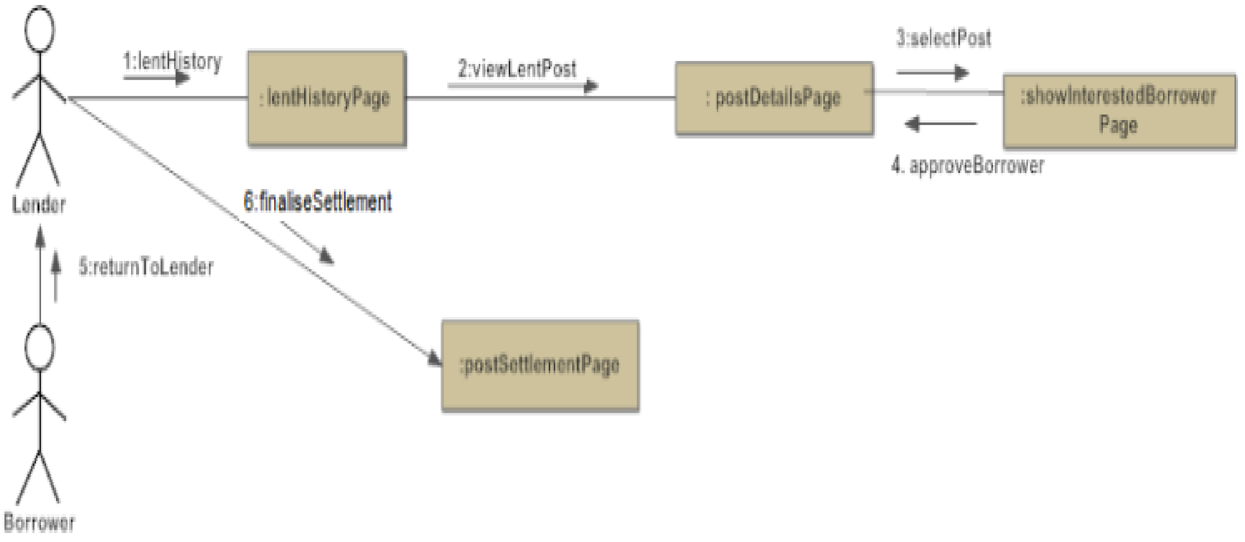


FIGURE 8.1.3 APPROVE BORROWER COLLABRATION DIAGRAM

8.2 SEQUENCE DIAGRAM

A Sequence diagram is an interaction diagram that shows how objects operate with one another and in what order. It is a construct of a message sequence chart. A sequence diagram shows object interactions arranged in time sequence.

8.2.1 Sequence diagram to create post

- If the user wants to lend his product,interacts with the CreatePostpage class which assists the user to create a post for his product.
- CreatePostpage triggers the PostCatalogue class to save the post details with a postID,after which successMsg is returned.
- After receiving the successMsg, the user can view the details of his post from PostPage.

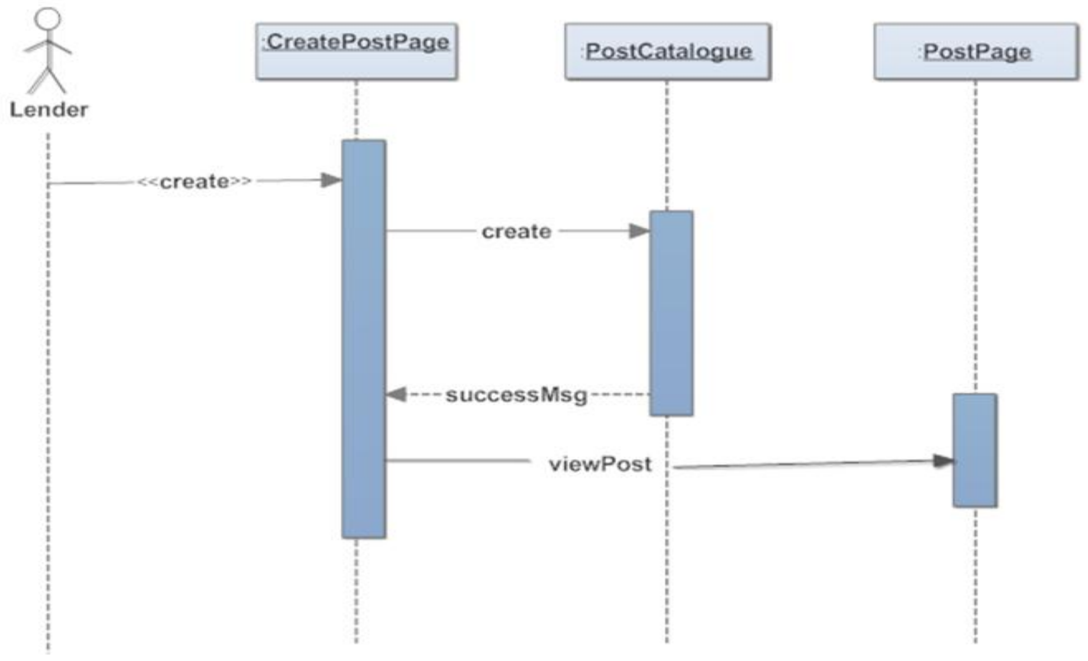


FIGURE 8.2.1 CREATE POST SEQUENCE DIAGRAM

8.2.2 Sequence diagram to borrow item

- If the user is interested in borrowing, interacts with the BorrowPage which displays the various products for borrowing
- From BorrowPage, Borrower can choose a particular category to view the products
- From ViewPostPage borrower can select a particular item and show interest in borrowing it
- From ShowInterestedBorrowerPage borrower can then add items to cart, once the product is added to the cart it will be redirected back to ViewpostPage.

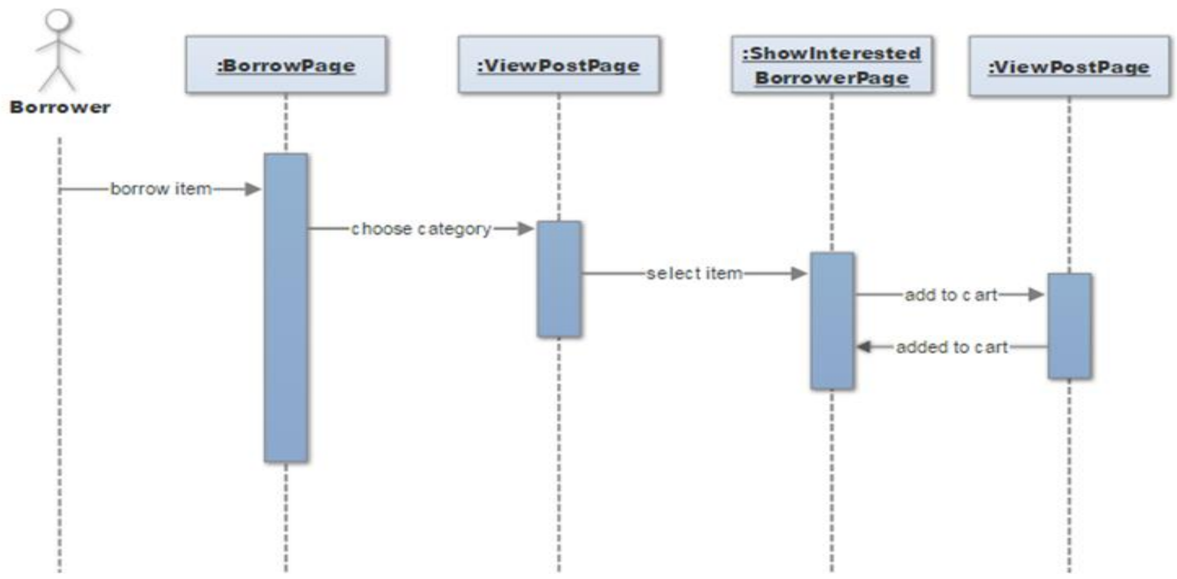


FIGURE 8.2.2 BORROW ITEM SEQUENCE DIAGRAM

8.2.3 Sequence diagram to approve post

- If lender is interested in seeing his lent history or wants to approve his product to be borrowed interacts with the LentHistoryPage
- LentHistoryPage displays all the lent posts in PostdetailsPage
- From PostdetailsPage, the lender will select the post for which he wants to approve the interested borrower to borrow
- From ShowIntrestedBorrowerPage, the lender will approve the interested borrower
- Once the borrower finalizes the payment and returns the product, the Lender settles up the transaction by clicking the settle up option on the PostSettlmentPage

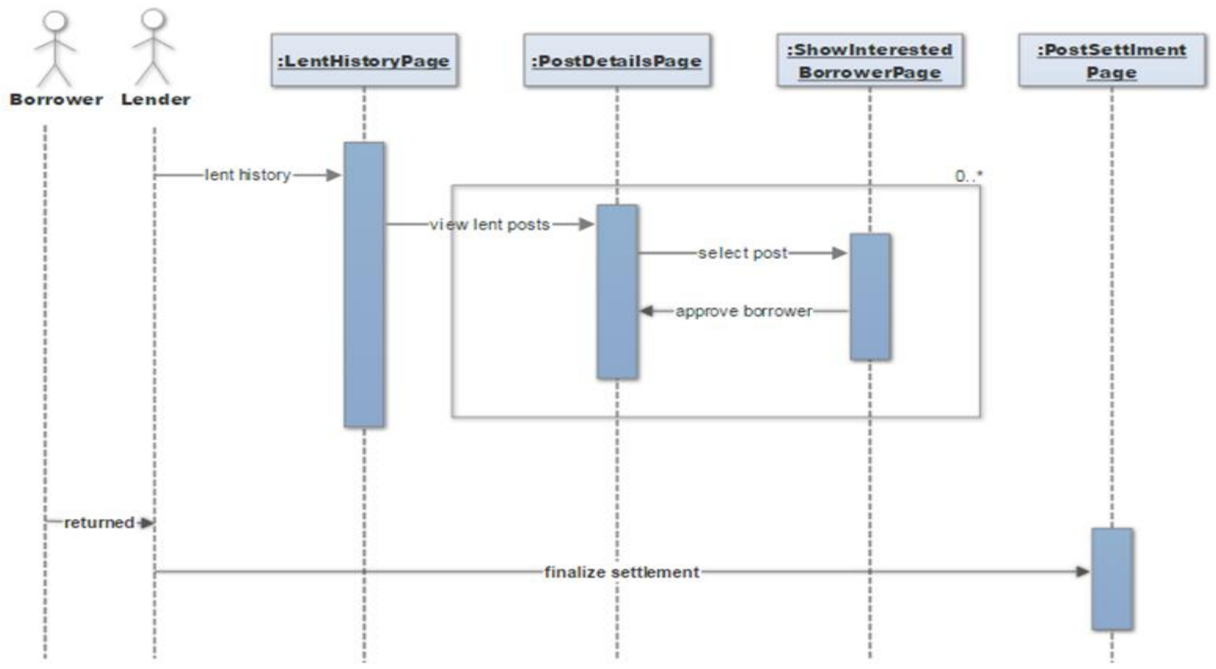


FIGURE 8.2.3 APPROVE POST SEQUENCE DIAGRAM

9. ACTIVITY DIAGRAM

In this section are activity diagrams that illustrate activities that take place while accessing the Lendkart System.

9.1 Logging in Lendkart

User, not registered previously, registers after which he is asked for login details. The details are validated and if valid details are provided User can perform Lend/Borrow tasks. If invalid details are provided system asks for correct credentials. Existing user is asked for details which are validated. If invalid details are provided system prompts for correct credentials.

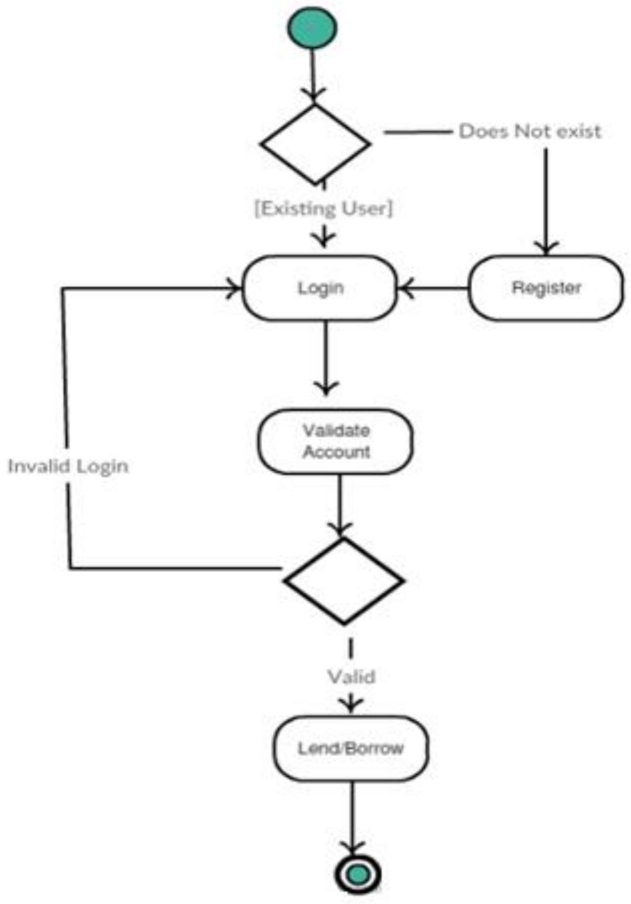


FIGURE 9.1 LOGIN ACTIVITY

9.2 Lending item

Lender can update profile by modifying his name contact and location details. Lender create posts of item by submitting the item title, description, category, images. Lender can see the post and if post has interested borrower .Lender can finish settlement if he has successfully received the amount for lent item. Lender can then rate the borrower.

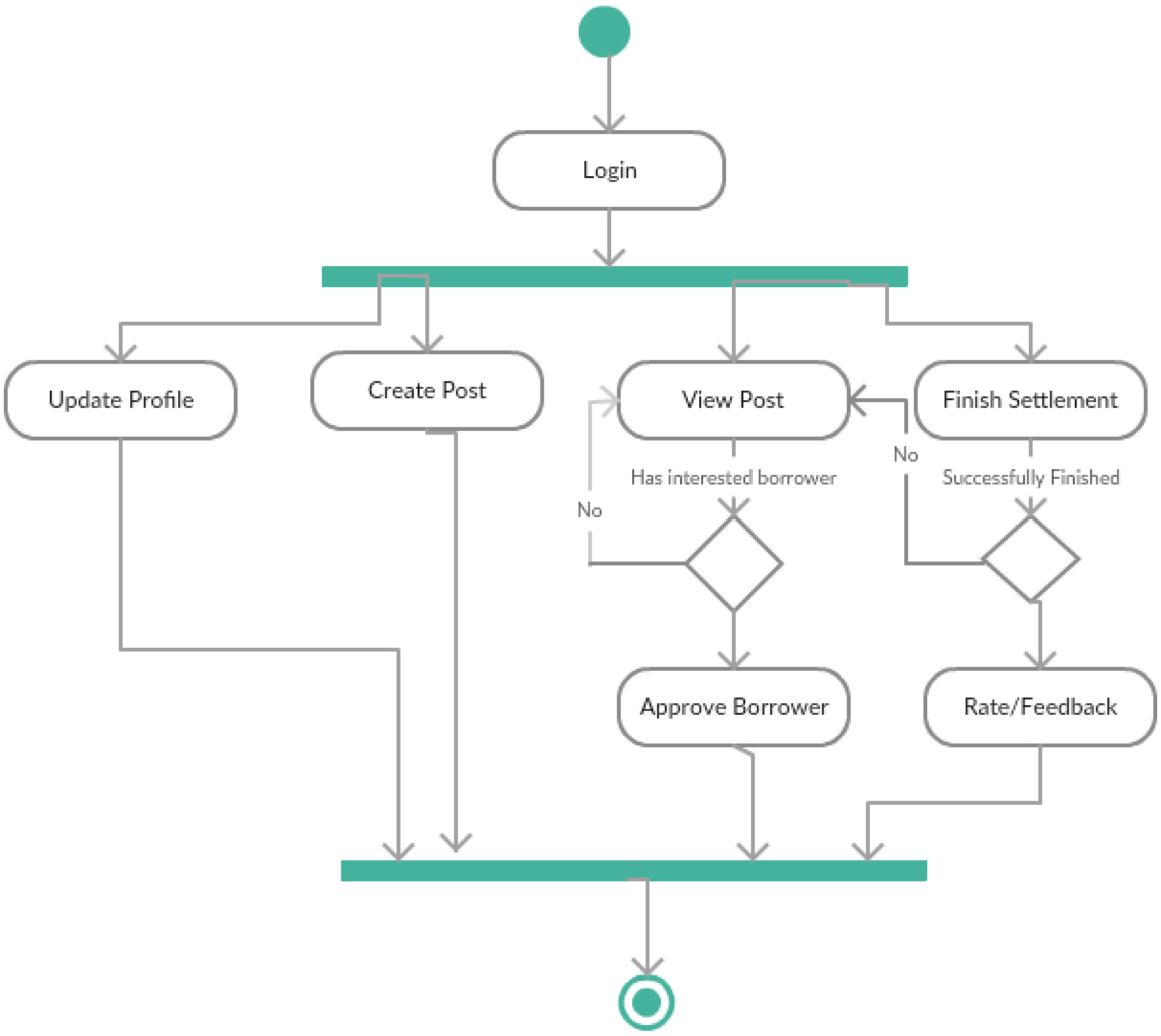


FIGURE 9.2 LENDING ACTIVITY

9.3 Borrowing item

Borrower can update his phone number and address through update profile. Borrower can search item post .If interested he can request an item for borrowing from the lender who issued the post. Also he can add item to wishlist through which he can see the items he was interested in later point of time.

If he requested the item for borrowing he cannot add it to the wish list. Borrower can see history of items to see the items was approved or not in which he was showed interest.

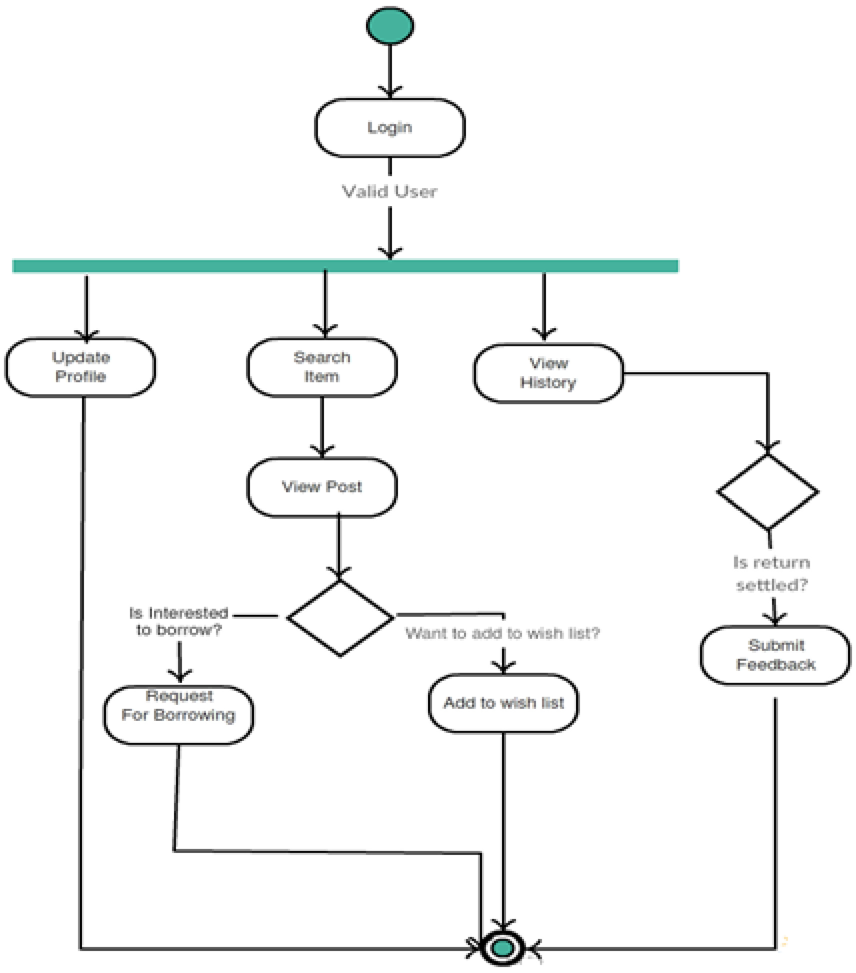


FIGURE 9.3 BORROW ACTIVITY

10. CLASS DIAGRAM

Classes are the set of objects that share the same attributes, operations, methods, relationships, and semantics. A good software engineering technique requires an engineer to divide up the system responsibilities into smaller sub-responsibilities. This separation process is beneficial because as a result of this process, the system gains many positive qualities such as reusability, modularity, and encapsulation.

Class diagram is the main building block of any object oriented solution. It shows all the classes in a system with their attributes and methods. It illustrates the relationship among various classes and also shows the interaction among various classes

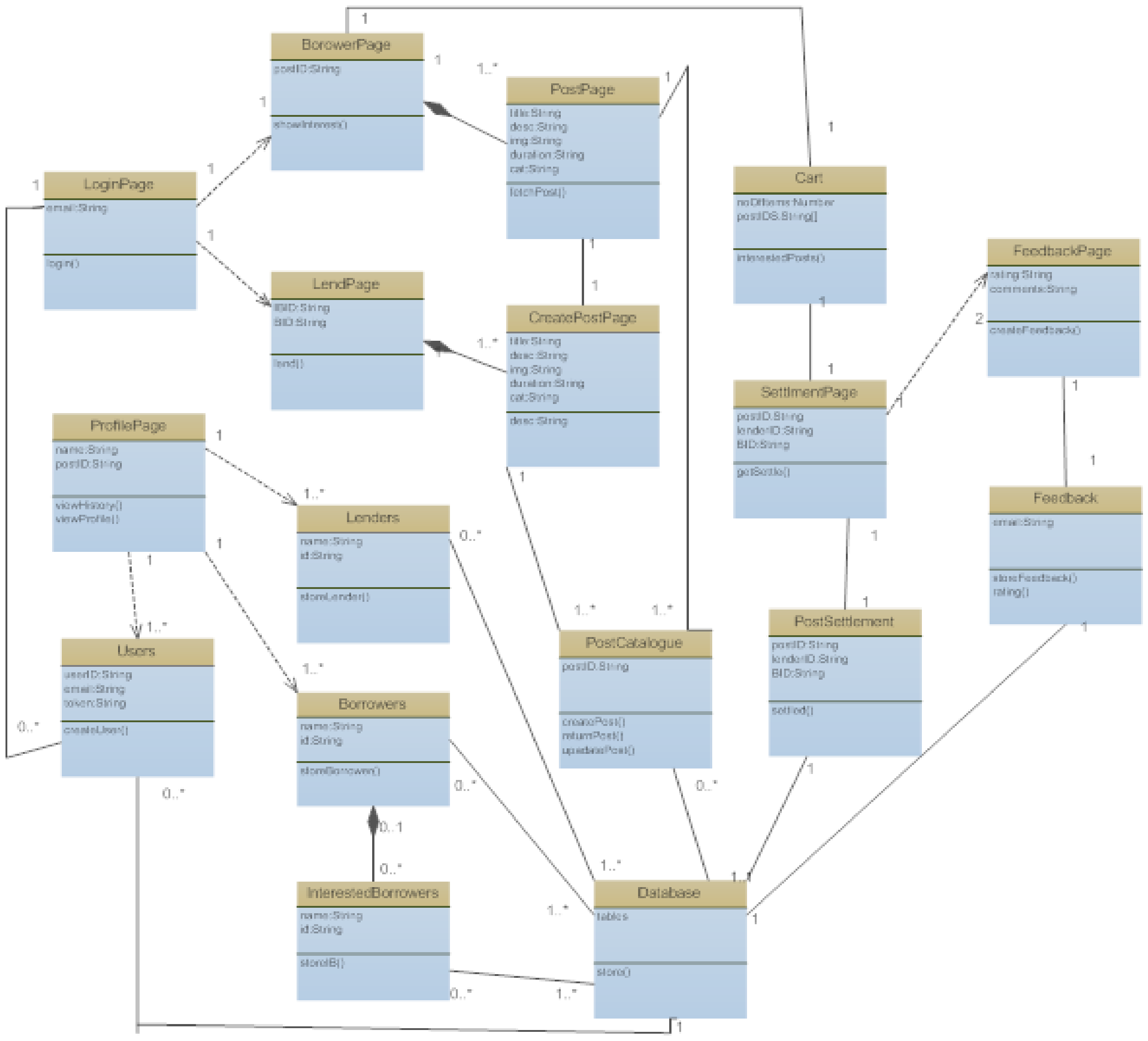


FIGURE 10 CLASS DIAGRAM

After determining all of the responsibilities of the system, a class is assigned to each of them so that each of the classes was in charge of certain responsibilities.

A class is represented in a class diagram by a rectangle having three parts as follows:

- Name of Class
- Attributes in the class
- Methods

Each class has been assigned separate responsibilities as per the Single Responsibility Principle. The responsibility of each class is listed below:

Class	Responsibility
LoginPage	*LoginPage* UI class assists the user in logging into the system. This class uses Google authentication API to authenticate the user
LendPage	*LendPage* UI class assists the user in entering the details about the items that he want to lend. It displays a form asking to enter information about the item that he want to lend.
BorrowPage	*BorrowPage* UI class is responsible for displaying the details about the various categories of the products.
Users	*Users* Model class contains the user details
PostPage	*PostPage* UI class is responsible for displaying the details of the post to the user. It also provides the functionalities like “show interest” to the user.
Cart	*Cart* is a model class to store the details of posts that the user has shown interest for.
SettlementPage	*SettlementPage* UI class is responsible marking the transaction over a post as settled.
PostSettlement	*PostSettlement* model class contains information related to transaction settlement for a post.
FeedbackPage	*FeedbackPage* UI class assists the user in providing feedback about the corresponding lender or borrower
Feedback	*Feedback* model class contains information about the feedback provided by the user.
ProfilePage	*ProfilePage* UI class displays the user’s profile information, it is also responsible for showing the history and wishlist options
InterestedBorrowers	*InterestedBorrowers* model class contains the information of the users who have shown interest to a particular post
Borrowers	*Borrowers* model class contains information of the users who have borrowed a particular post
PostCatalogue	*PostCatalogue* model class contains all the information of the created posts
Wishlist	*Wishlist* model class contains the information of the posts which were marked favorite by the user
Database	*Database* class contains the information of the database connections

TABLE 10 CLASS RESPONSIBILTIY

The relationships among the classes are as follows:

- *LoginPage* class assists the user in logging into the system. This class uses Google authentication API to authenticate the user. Once authentication is done successfully, user is redirected to *home page .* If the user wants to create a post to lend an item, he selects the Lend tab. LendPage asks user to enter information about the product that he wants to lend. Once the details of the item are successfully saved,the information of the created post is stored in the *PostCatalogue* model class. *PostPage* class displays the complete information of the post.
- BorrowPage class assists the user in borrowing items with the help of LendKart. It is responsible for displaying the various categories of items available.
- User can view the details of a particular post with the help of the UI class named *PostPage*. If the user wants to borrow an item, then he can show interest in that particular post. Lender of that particular post will be informed about the same. Also, that specific Ad is added to user's *CART.* If the user likes the item but does not want to send interest to borrower, then he can add the item to his *WishList.*
- *SettlementPage* class handles the transaction information about a particular item. This class interacts with *Feedback* class that assists the user in providing feedback for the corresponding lender or borrower.
- *ProfilePage* handles the information about the user. User can update his profile, view his history which will redirect to the Lend history and Borrow history. From Lend history, the lender can approve the interested borrowers to borrow his product.

11. PACKAGING

Packaging is an additional level of organization, which allows further development of system architecture layers. By grouping objects that interact most with each other into one package, there is less interfacing between packages. This process of analyzing the connections between objects and classes increases the cohesion and lowers the coupling that is involved between the presentation, application domain and logic, and storage layers.

11.1 Three Tier Architecture

The package diagram demonstrates elements into organized packages. Usually there are horizontal layers and vertical partitions to arrange the objects in the program. In Figure 10, there are three horizontal layers, which are the following: Presentation, Application, and Storage. The presentation layer contains the GUI, which will be the interface between the user and the system's internal functions. The application layer consists of domain and service layer objects that execute the main business logic such as listening of packets and sending messages. Finally the storage layer keeps the log file separate from the remainder of the system.

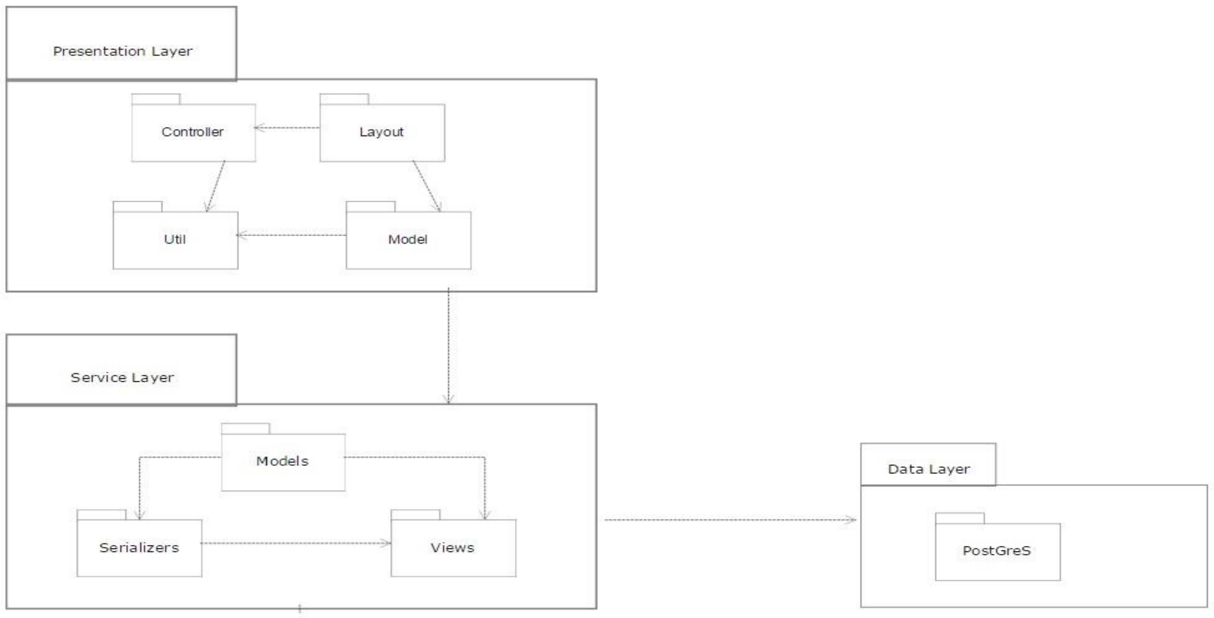


FIGURE 11.1 PACKAGE DIAGRAM

11.2 Presentation Layer

The GUI in the presentation layer is provided to user through android interface.

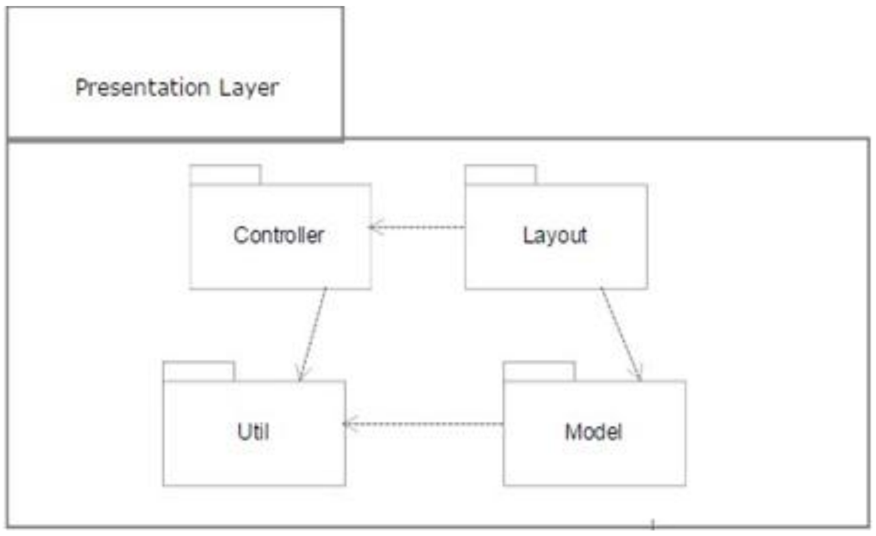


FIGURE 11.2 PRESENTATION LAYER PACKAGE

11.3 Service Layer

This layer pertains to the service provided to front end from django backend. The web services are provided through mvt framework of Django
In mvt, a request to a URL is dispatched to a View. This View calls into the Model, performs manipulations and prepares data for output. The data is passed to a Template that is rendered an emitted as a response. Ideally in web frameworks, the controller is hidden from view.

This is where the difference is from MVC: in mvc, the user interacts with the GUI, the controller handles the request and notifies the model and the view queries the model to display the result to the user.

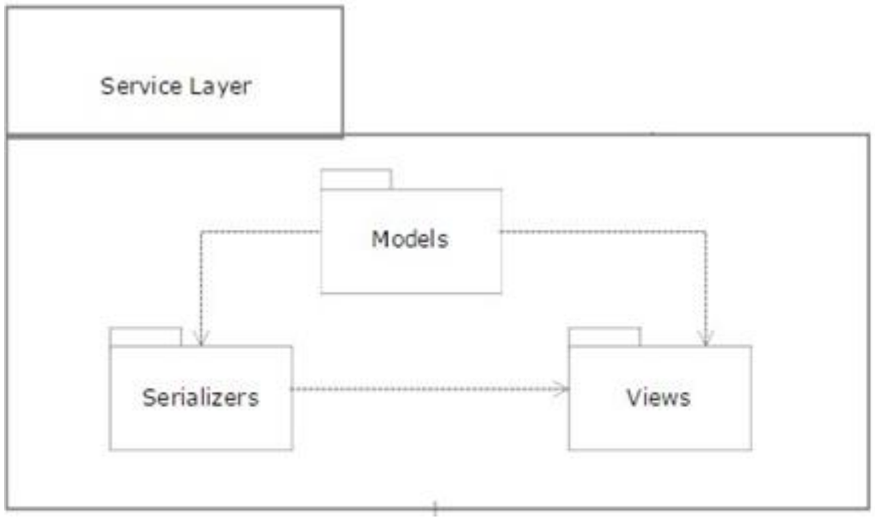


FIGURE 11.3 SERVICE LAYER PACKAGE

11.4 Data Layer

This layer divides the storage capabilities that this program requires from the remainder of the system. This layer contains the data received and sent from android to django backend.

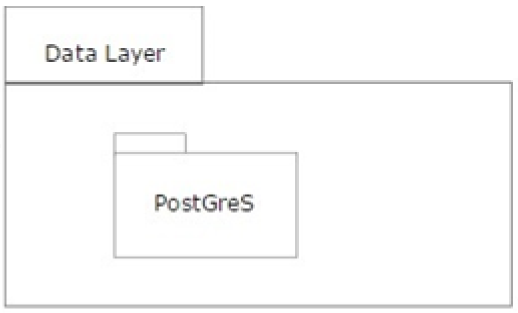


FIGURE 11.4 DATA LAYER PACKAGE

12. COMPONENT DIAGRAM

Component diagrams are used to model physical aspects of a system. Physical aspects are the elements like executables, libraries, files, documents etc which resides in a node. Component diagrams are used to visualize the organization and relationships among components in a system.

Lendkart has the following components:

1. **Android UI**: The Android Studio is used to design the front end UI, and the app is deployed on the user's smart phone.
2. **Google authentication system**: The sign in for the app is done through the Gmail. So, the Google authentication services are used to authenticate the Lendkart users.
3. **PostgreSQL database**: The complete information about the lenders, borrowers, posts, feedbacks, rating etc are stored in the PostgreSql database.
4. **Image repository**: The pictures of the products that are being lent and borrowed are stored on a third party repository. Amazon S3 is used for storing the images
5. **Django webservices**: The webservices are coded in python language in Django framework. Django is a MVC framework where the data is sent and rendered in the form of JSON.

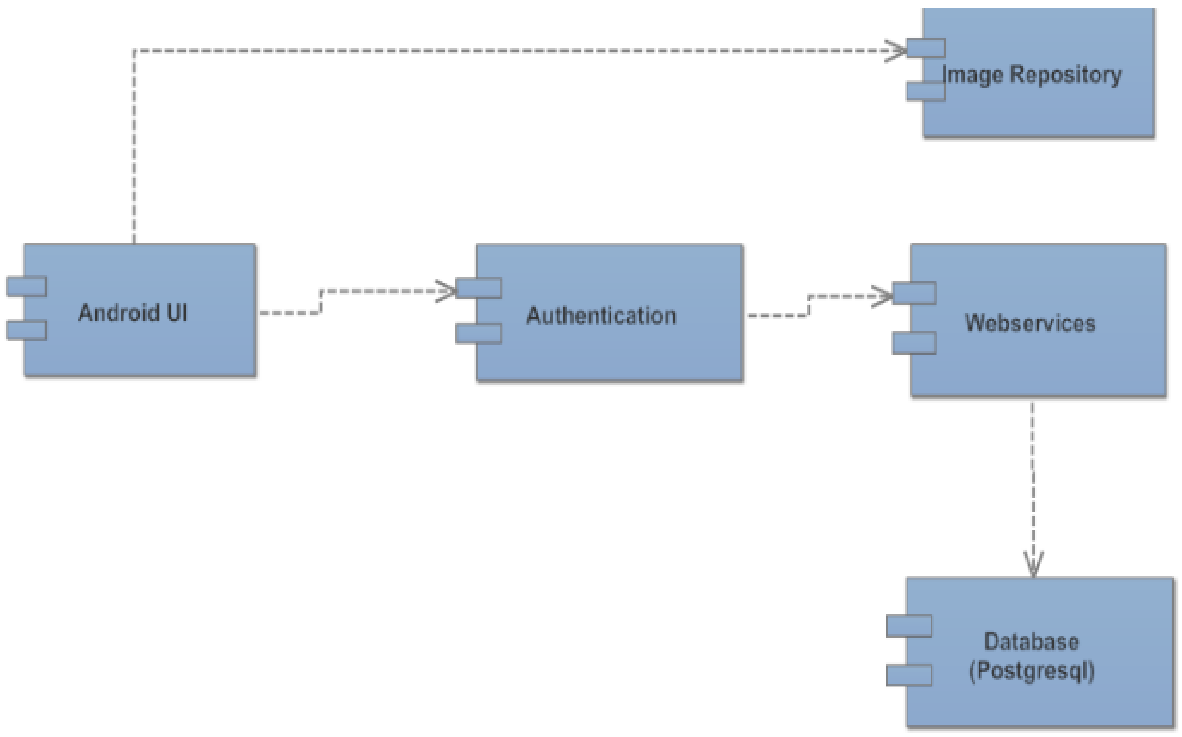


FIGURE 12 COMPONENT DIAGRAM

13. DEPLOYMENT DIAGRAM

Deployment diagrams are used to visualize the topology of the physical components of a system where the software components are deployed. Deployment diagrams are used for describing the hardware components where software components are deployed.

- The Lendkart.apk is deployed on the android mobile phones
- The images uploaded in this application are stored in an external storage ie in the the Amazon S3 cloud
- The django REST framework services are used for managing the requests from the client and responses from the server
- The requests and the responses are sent in the form of JSON data
- The webservices are deployed on the AWS EC2(Amazon web services elastic compute cloud) instance

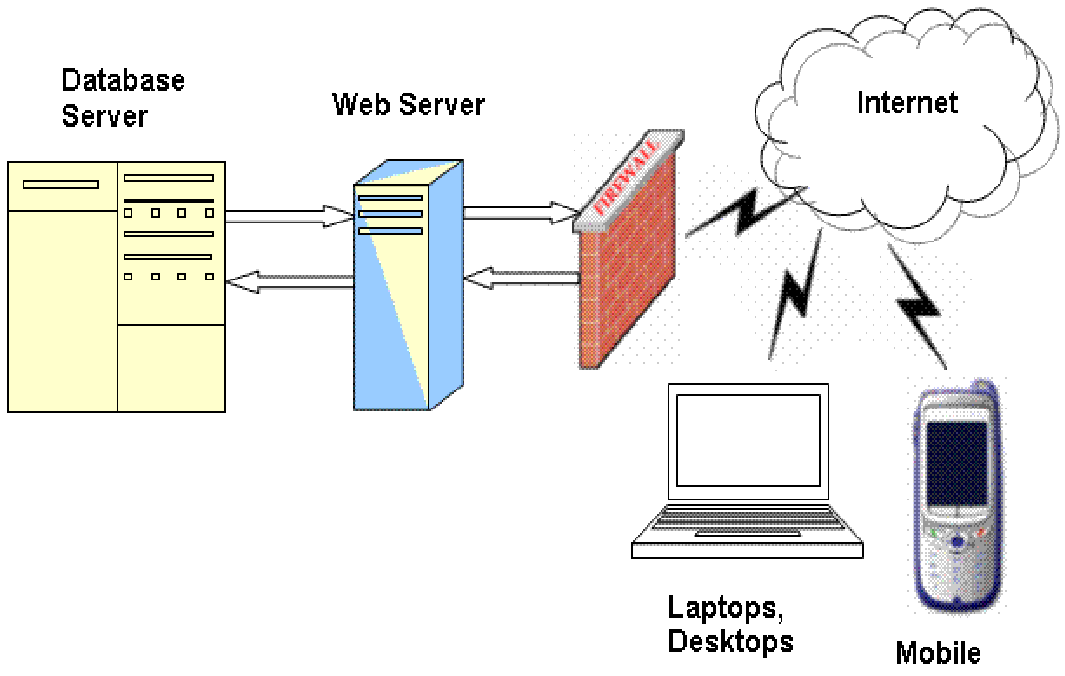
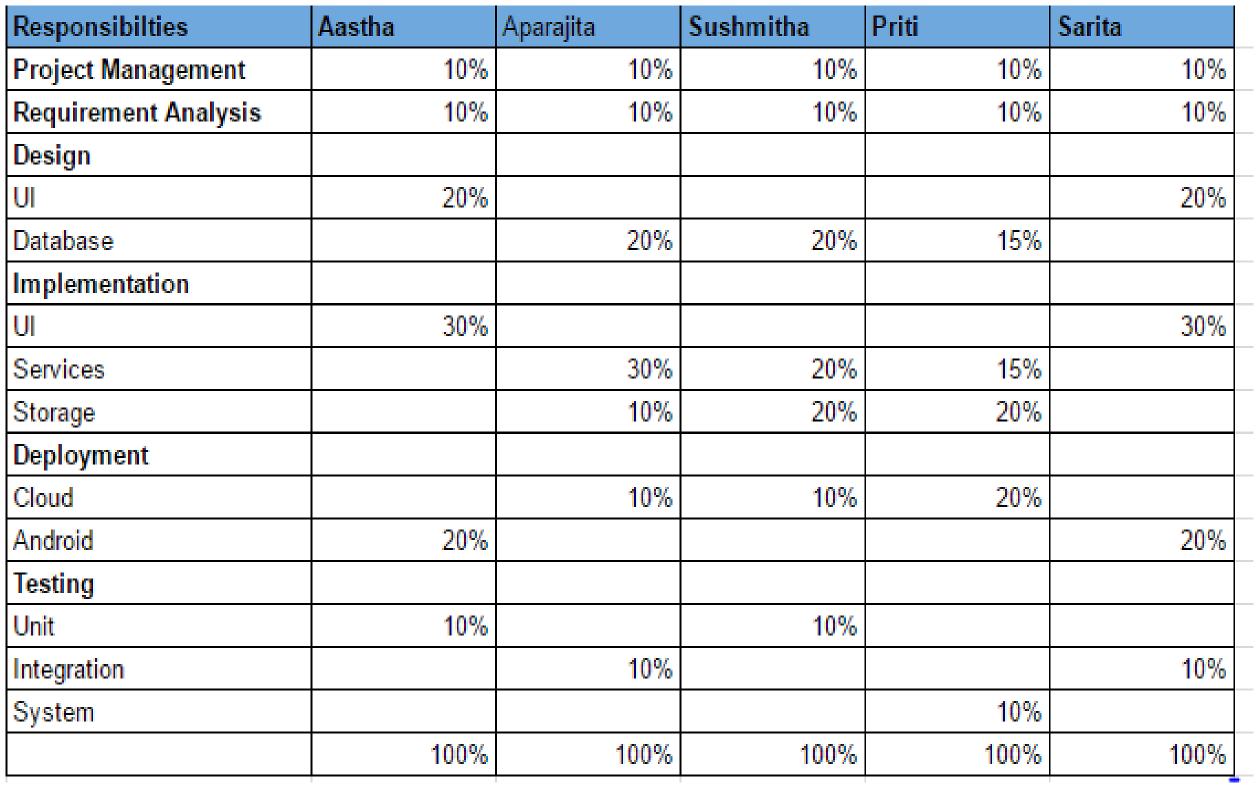


FIGURE 13 DEPLOYMENT DIAGRAM

14. Administrative Information

Below table shows each member contribution to the LendKart application.



Responsibilties	Aastha	Aparajita	Sushmitha	Priti	Sarita
Project Management	10%	10%	10%	10%	10%
Requirement Analysis	10%	10%	10%	10%	10%
Design					
UI	20%				20%
Database		20%	20%	15%	
Implementation					
UI	30%				30%
Services		30%	20%	15%	
Storage		10%	20%	20%	
Deployment					
Cloud		10%	10%	20%	
Android	20%				20%
Testing					
Unit	10%		10%		
Integration		10%			10%
System				10%	
	100%	100%	100%	100%	100%

15. TABLE CONTIRBUTION

15. PROBLEMS, CONFLICTS, AND RESOLUTION

- The team faced many challenges while developing the project as many new technologies has been used. Team struggled python version compatibility with django, framework used for service layer building.
- Android studio is quite a dynamic tool and version compatibility was a big issue. Each time migration of code built on one system is done, errors and warnings will be thrown. At same time studio needs huge amount of memory. Thus system performance also degraded whenever it is used.
- Integration of front end with backend was also a major task. Because LendKart almost all the functionalities talk to database for data display. Thus integration was challenging task and handling large number of REST call and responses.
- Image, resultant of creating post on application, has to be uploaded to server and being a media file bandwidth congestion and management of the same in the backend was a big challenge. Thus after any trial and error we came up with Amazon S3 which is simple storage service in internet and is compatible with web services using REST.

- Another major challenge is to access the database. Initially a personal computer was set up as server to hit the web services but the team faced different issues such a system heat up, network issues etc. Thus deployment on cloud to access the services without any hindrance. But finding one server and deployment was also not an easy task. We went through many available options and it took ample of time for the same. Finally after tedious search deployment of database is done on server i.e AWS services.