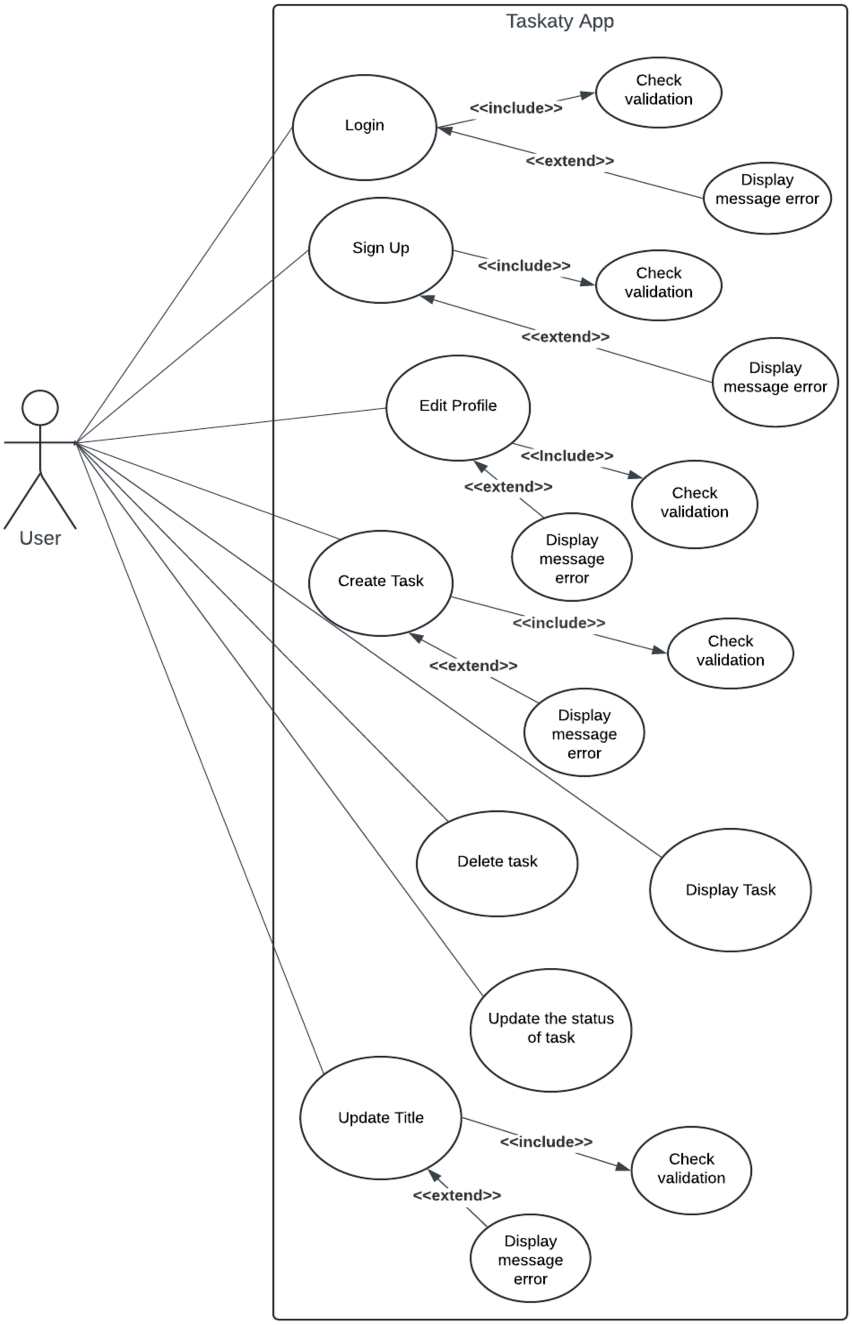
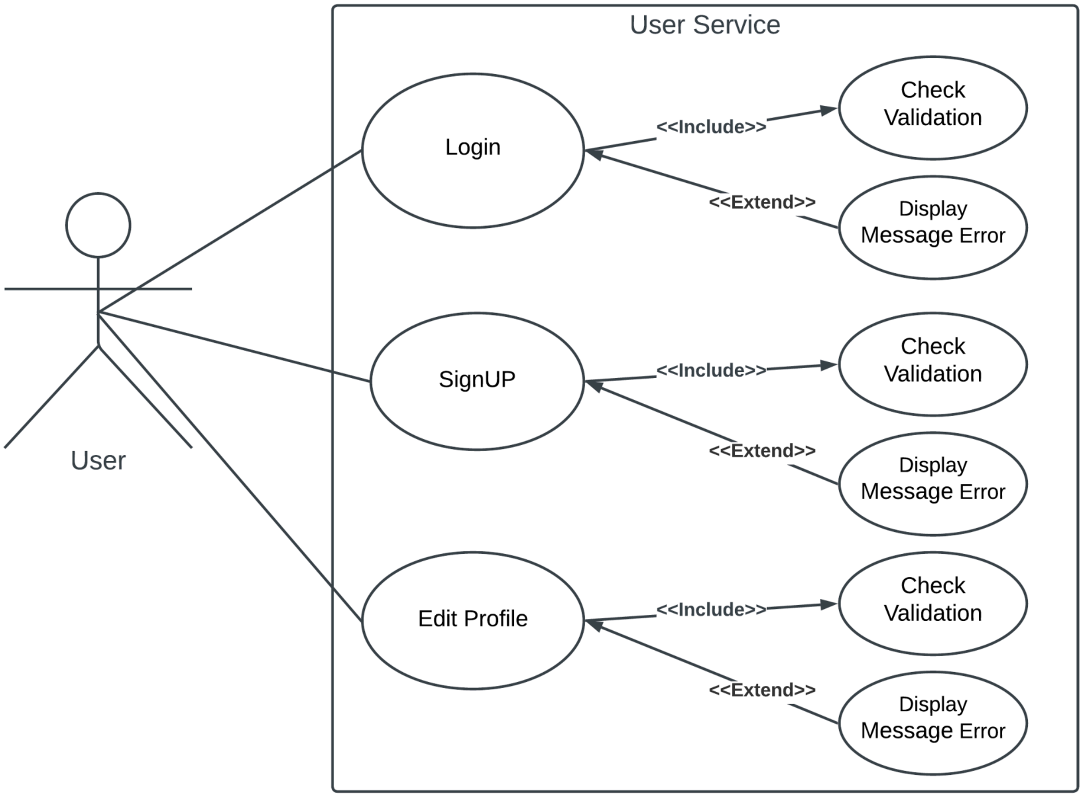
Software Requirements Specification (SRS)

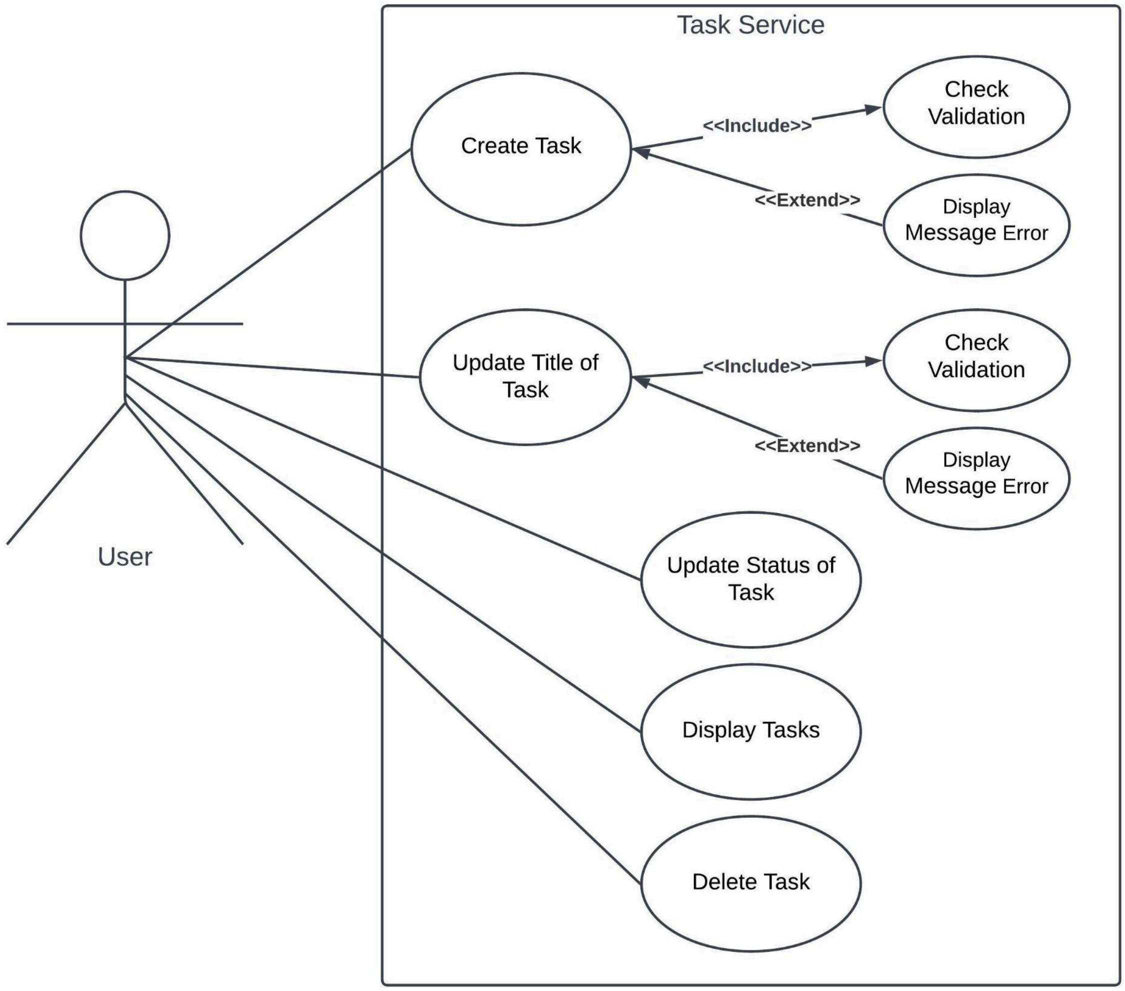
Use Case



User Service :



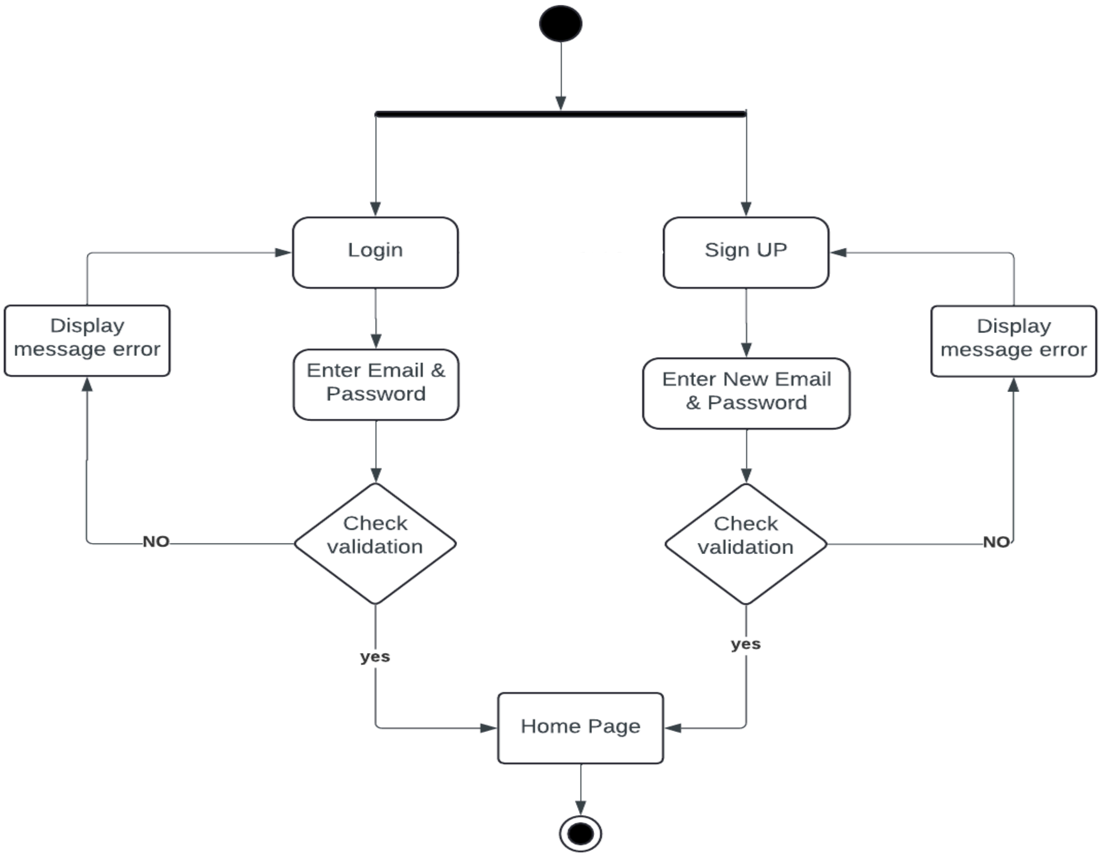
Task Service :



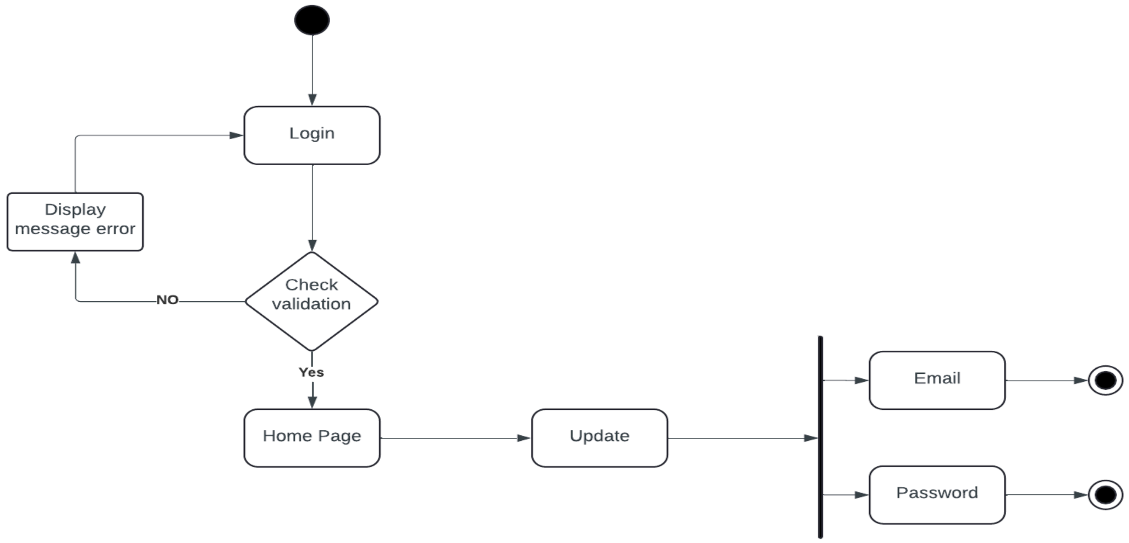
Activity diagram

User Service :

- **Login & Signup**

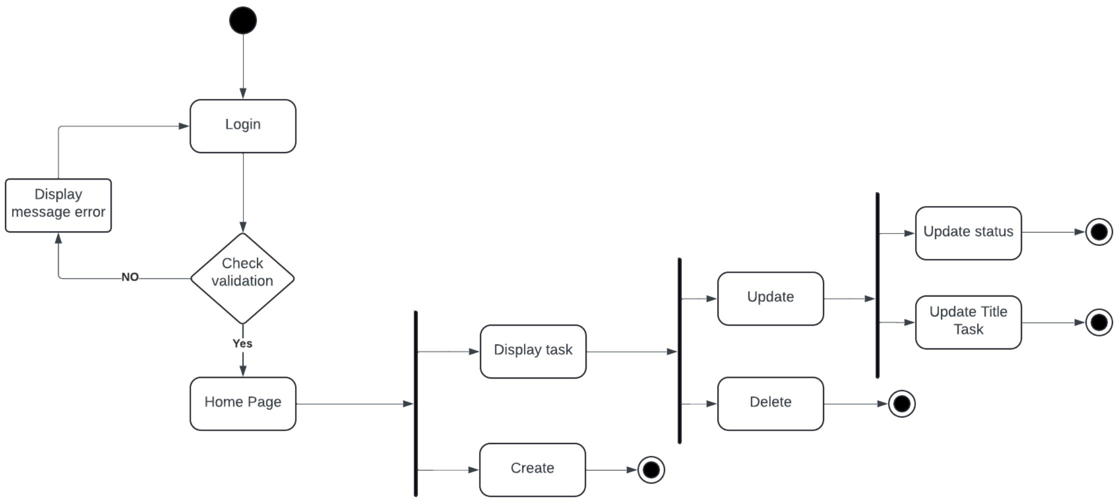


- **Edit Profile(Email,Password)**



Task Service :

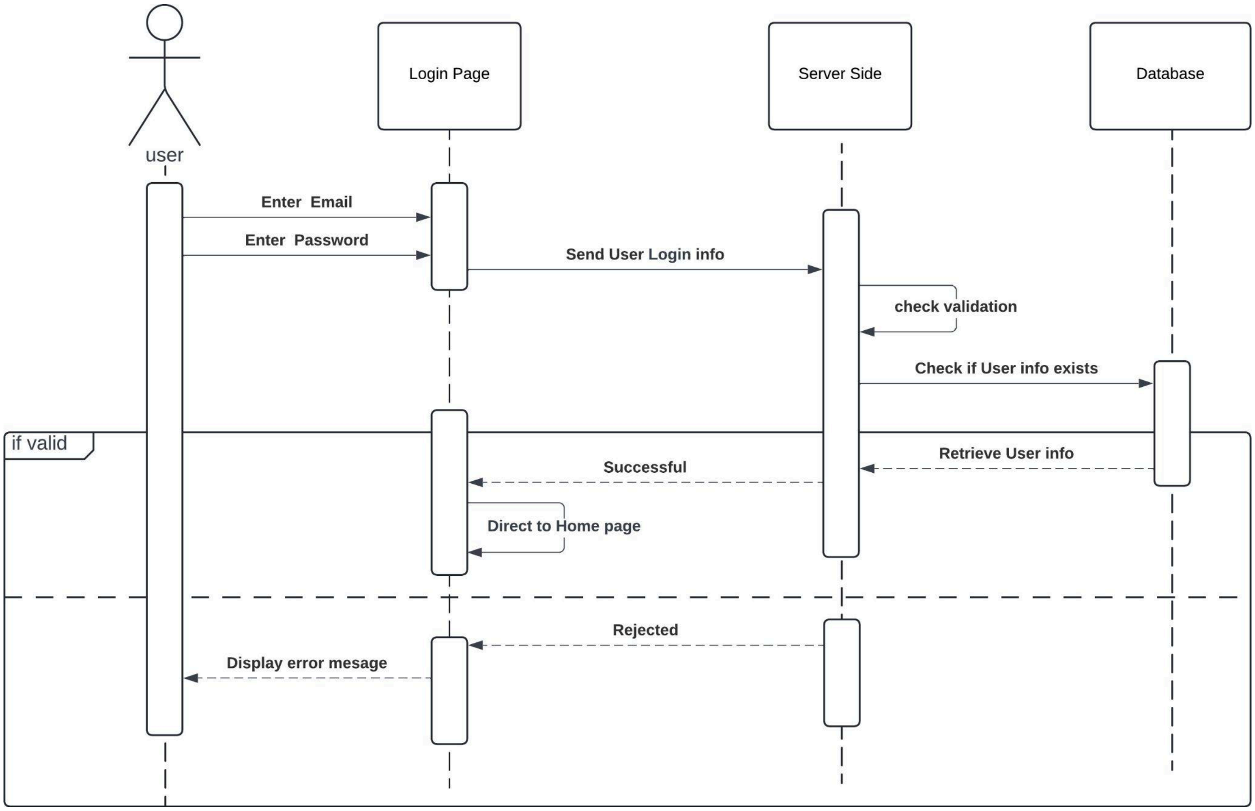
- **Task Management (Create - Display - Update - Delete)**



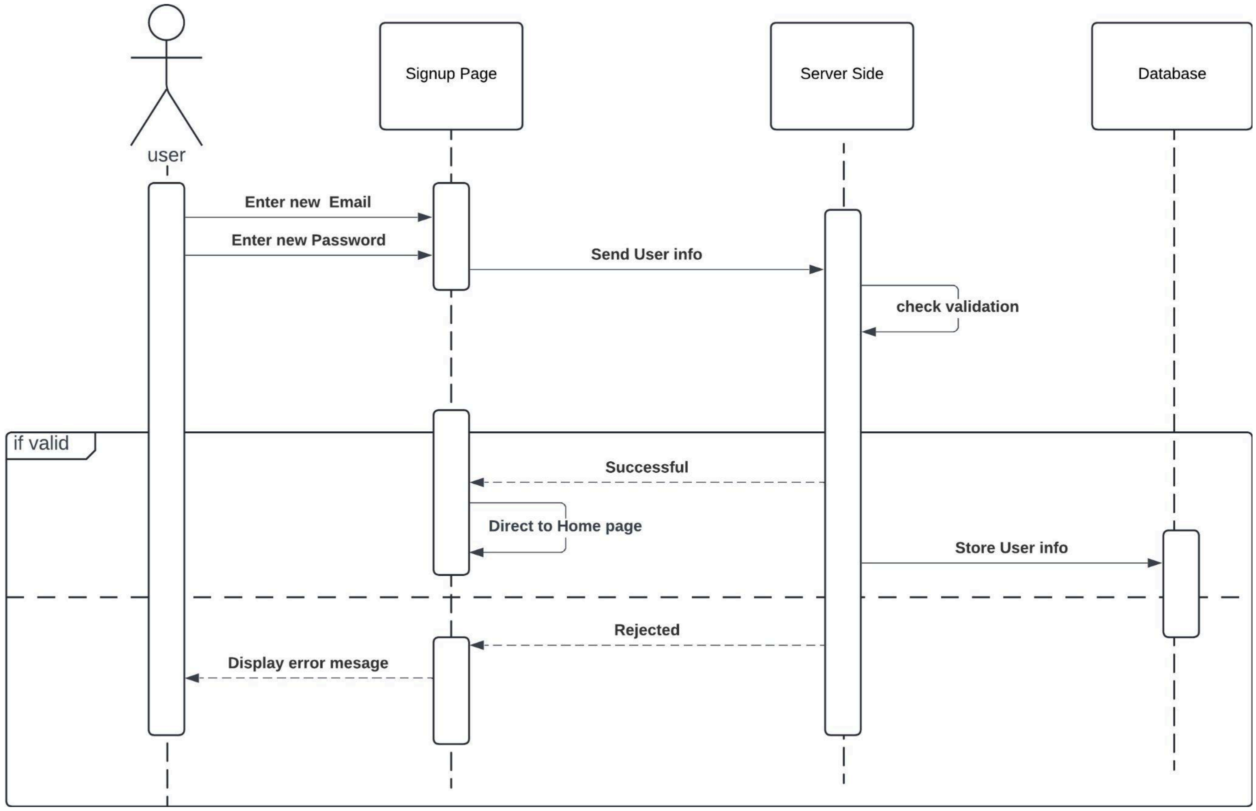
Sequence Diagram

User Service :

Login :

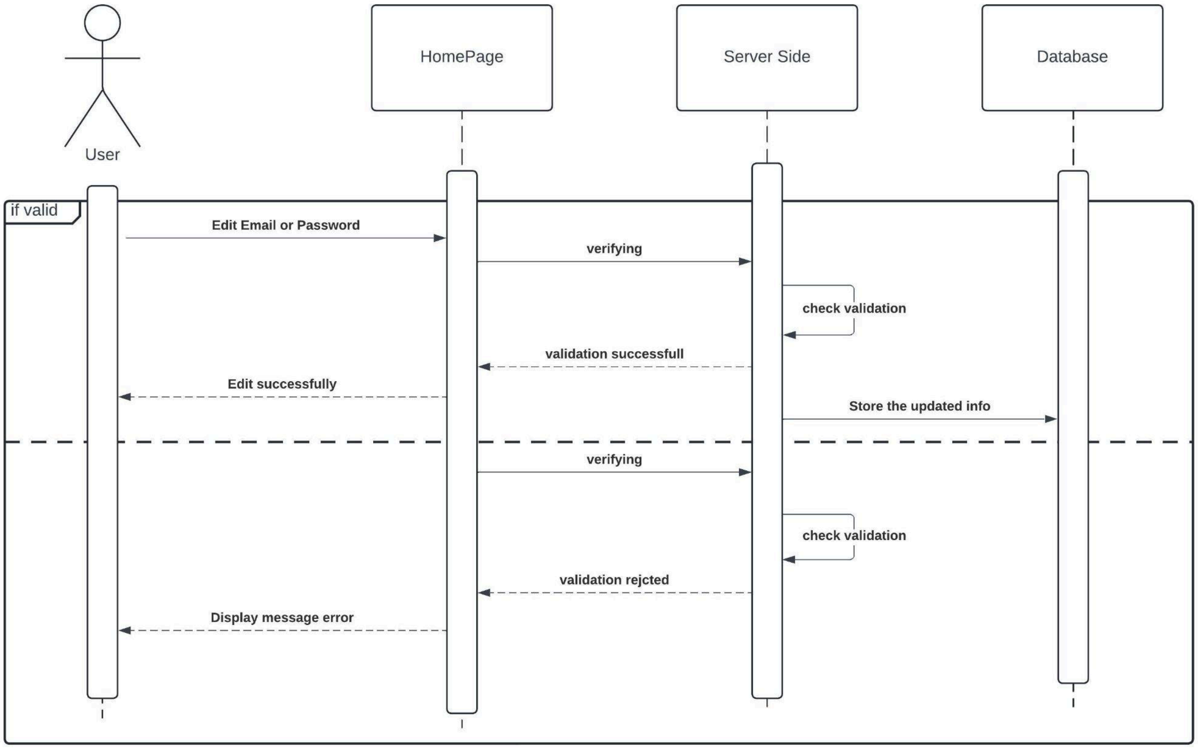


Sign Up :

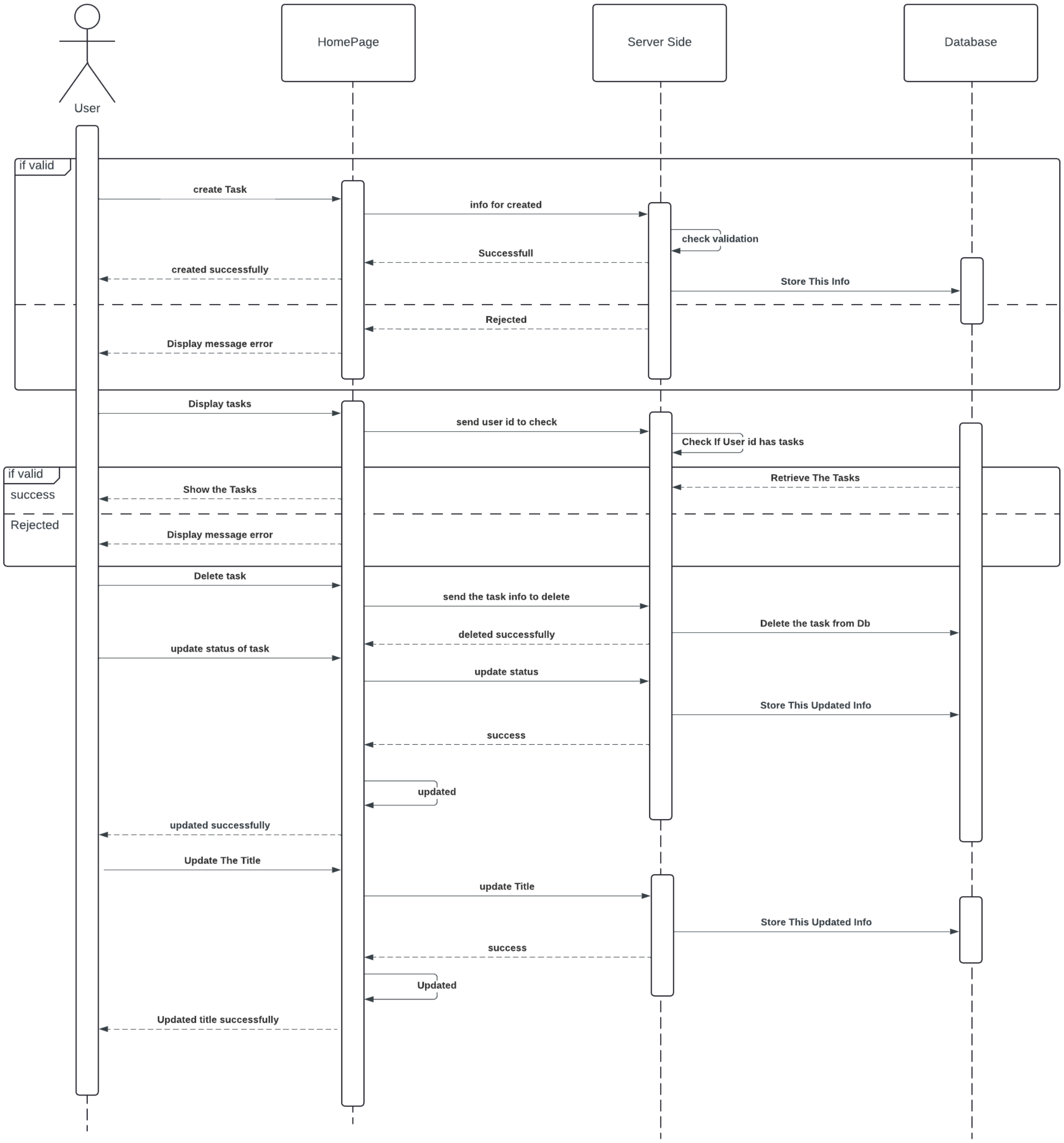


user
Signup Page
Server Side
Database
Enter new Email
Enter new Password
Send User info
check validation
if valid
Successful
Direct to Home page
Store User info
Rejected
Display error mesage

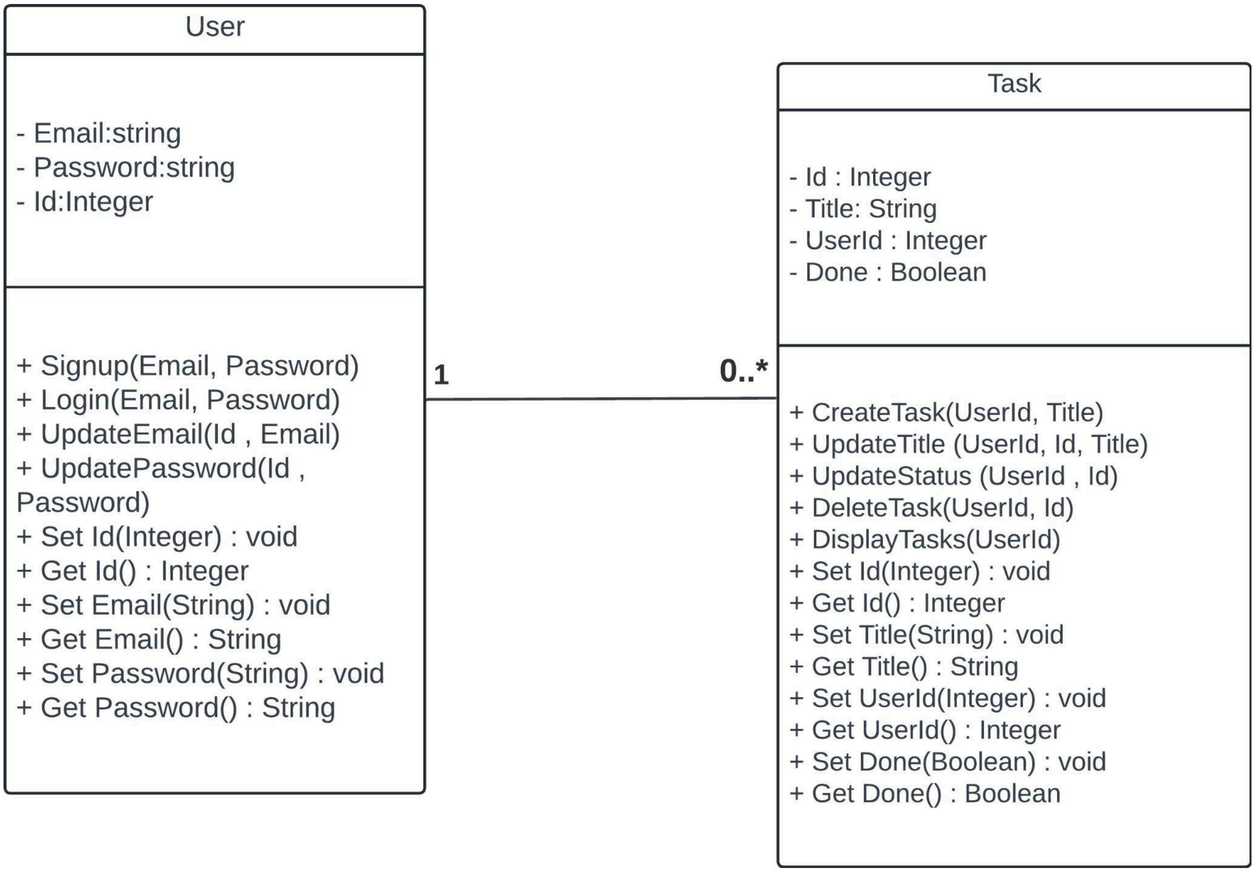
Edit Profile :



Task Service (Create - Display - Update - Delete) :



Class Diagram



OCL Diagram

Task Microservice:

Constraint 1: Tasks must have a title.

context Task
invariant: self.title <> null

Constraint 2: Task titles must be between 1 and 100 characters long.

context Task
invariant: self.title.size() >= 1 and self.title.size() <= 100

Constraint 3: Tasks must have a user ID associated with them.

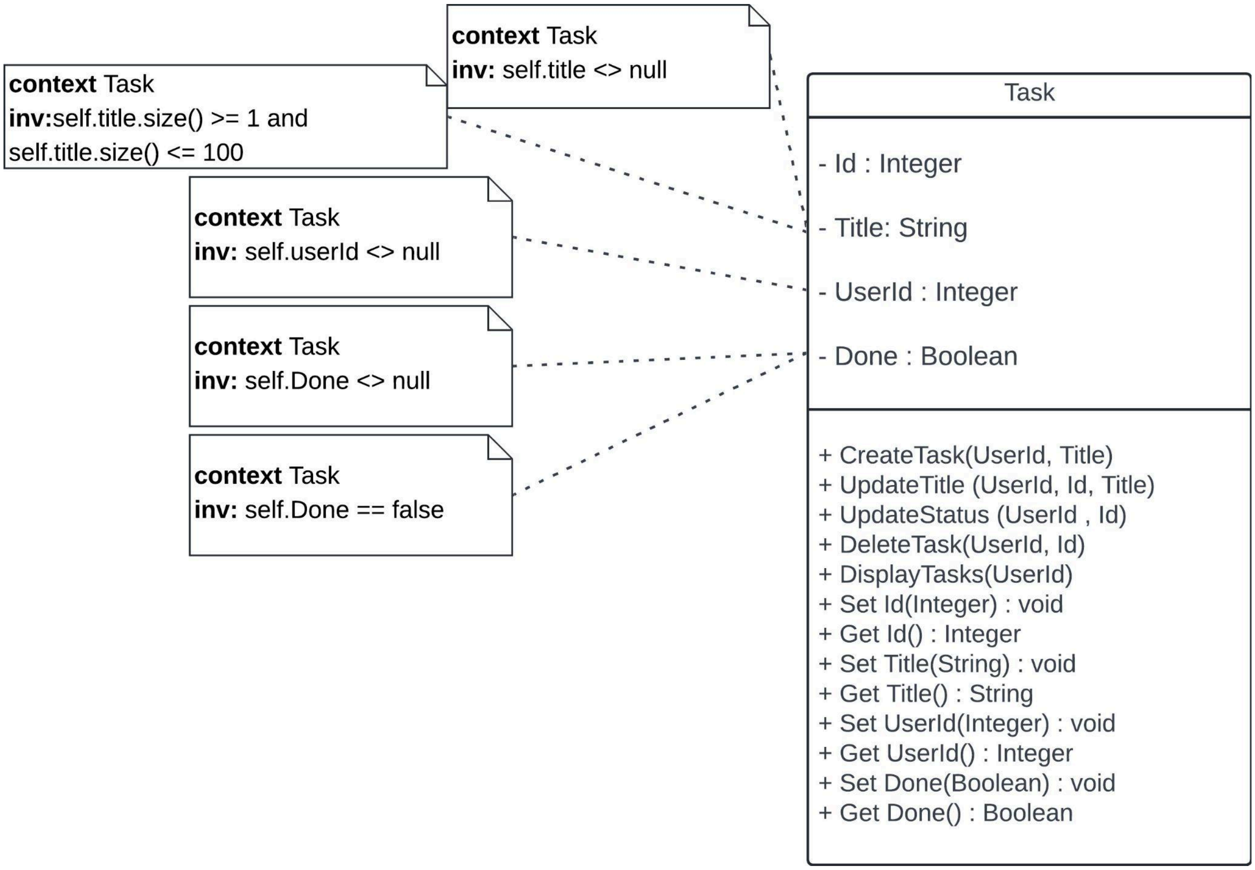
context Task
invariant: self.userId <> null

Constraint 4: Tasks must have a Status.

context Task
invariant: self.Done <> null

Constraint 5: Tasks when created must have a default status value.

context Task
invariant: self.Done == false



context Task
inv: self.title <> null

context Task
inv:self.title.size() >= 1 and
self.title.size() <= 100

context Task
inv: self.userId <> null

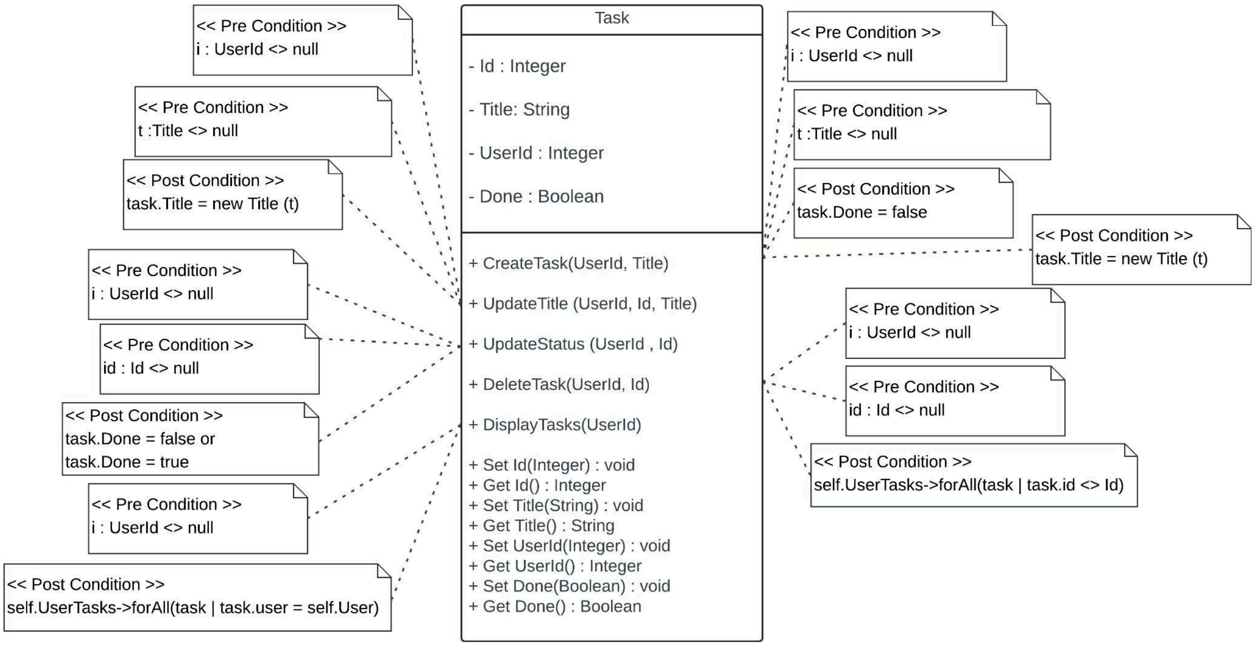
context Task
inv: self.Done <> null

context Task
inv: self.Done == false

Task

- Id : Integer
- Title: String
- UserId : Integer
- Done : Boolean

+ CreateTask(UserId, Title)
+ UpdateTitle (UserId, Id, Title)
+ UpdateStatus (UserId , Id)
+ DeleteTask(UserId, Id)
+ DisplayTasks(UserId)
+ Set Id(Integer) : void
+ Get Id() : Integer
+ Set Title(String) : void
+ Get Title() : String
+ Set UserId(Integer) : void
+ Get UserId() : Integer
+ Set Done(Boolean) : void
+ Get Done() : Boolean


Task
- Id : Integer
- Title: String
- UserId : Integer
- Done : Boolean
+ CreateTask(UserId, Title)
+ UpdateTitle (UserId, Id, Title)
+ UpdateStatus (UserId , Id)
+ DeleteTask(UserId, Id)
+ DisplayTasks(UserId)
+ Set Id(Integer) : void
+ Get Id() : Integer
+ Set Title(String) : void
+ Get Title() : String
+ Set UserId(Integer) : void
+ Get UserId() : Integer
+ Set Done(Boolean) : void
+ Get Done() : Boolean
<< Pre Condition >>
i : UserId <> null
<< Pre Condition >>
t :Title <> null
<< Post Condition >>
task.Title = new Title (t)
<< Pre Condition >>
i : UserId <> null
<< Pre Condition >>
id : Id <> null
<< Post Condition >>
task.Done = false or
task.Done = true
<< Pre Condition >>
i : UserId <> null
<< Post Condition >>
self.UserTasks->forAll(task | task.user = self.User)
<< Pre Condition >>
i : UserId <> null
<< Pre Condition >>
t :Title <> null
<< Post Condition >>
task.Done = false
<< Post Condition >>
task.Title = new Title (t)
<< Pre Condition >>
i : UserId <> null
<< Pre Condition >>
id : Id <> null
<< Post Condition >>
self.UserTasks->forAll(task | task.id <> Id)

User Microservice:

Constraint 1: User email addresses must be unique within the system.

context User

invariant: User.allInstances()->forAll(u1, u2 | u1 <> u2 implies u1.email <> u2.email)

Constraint 2: User must have an email address.

context User

invariant: self.email <> null

Constraint 3: User email addresses must follows the standard email format.

context User

invariant:self.email.matches('[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\\.[a-zA-Z]{2,}')

Constraint 4: User must have a Password.

context User

invariant: self.passowrd <> null

Constraint 5: User Password size must be between (6-20).

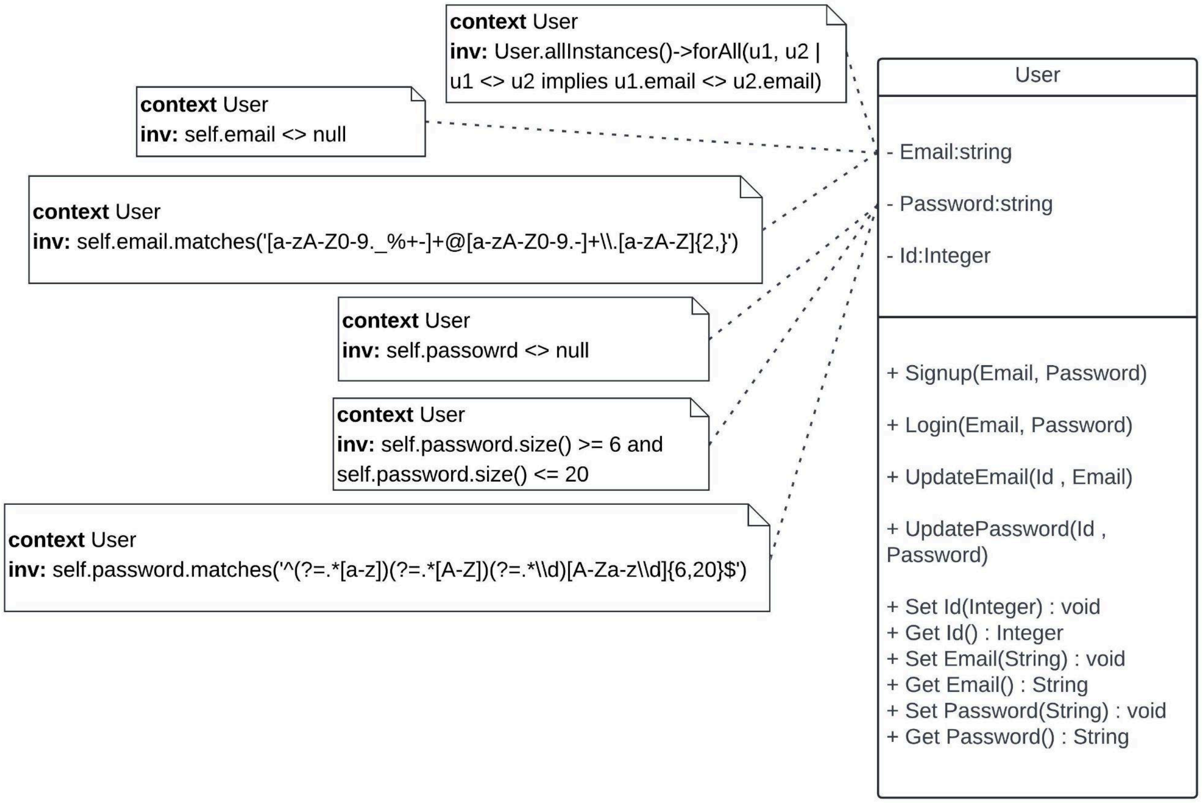
context User

invariant: self.password.size() >= 6 and self.password.size() <= 20

Constraint 6: User Passwords must meet some complexity requirements and regular expressions.

context User

invariant:self.password.matches('^(?=.*[a-z])(?=.*[A-Z])(?=.*\\d)[A-Za-z\\d]{6,20}$')



context User
inv: User.allInstances()->forAll(u1, u2 |
u1 <> u2 implies u1.email <> u2.email)

context User
inv: self.email <> null

context User
inv: self.email.matches('[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\\.[a-zA-Z]{2,}')

context User
inv: self.passowrd <> null

context User
inv: self.password.size() >= 6 and
self.password.size() <= 20

context User
inv: self.password.matches('^(?=.*[a-z])(?=.*[A-Z])(?=.*\\d)[A-Za-z\\d]{6,20}$')

User

- Email:string

- Password:string

- Id:Integer

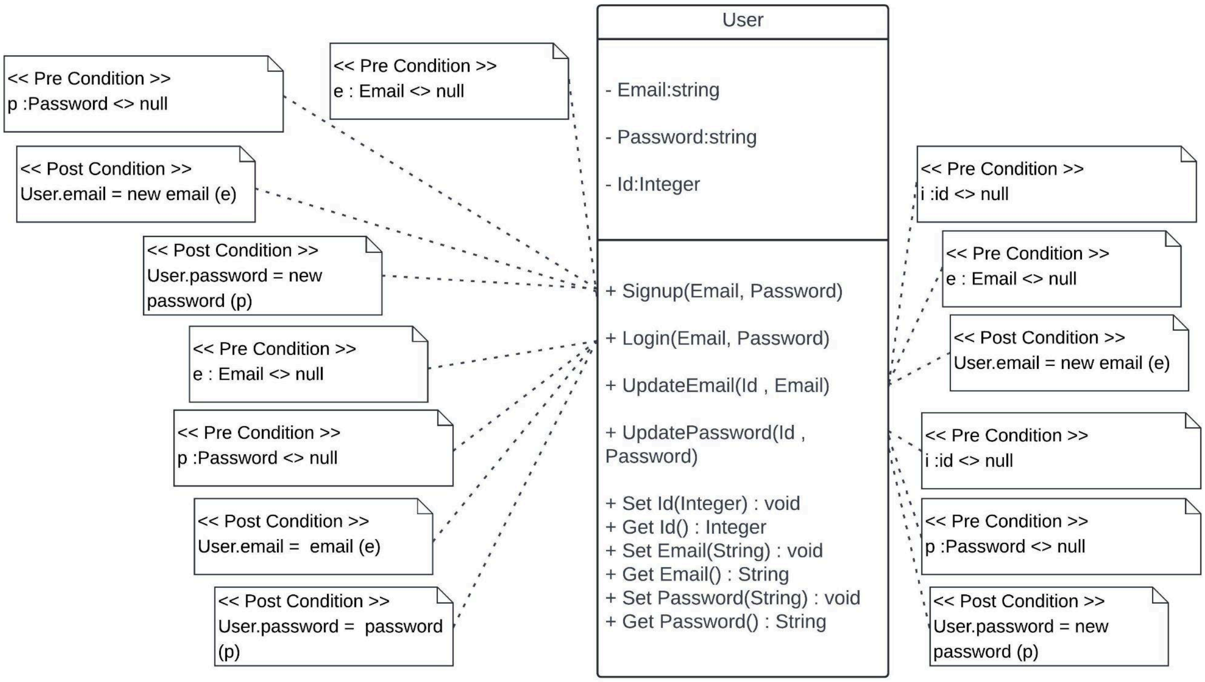
+ Signup(Email, Password)

+ Login(Email, Password)

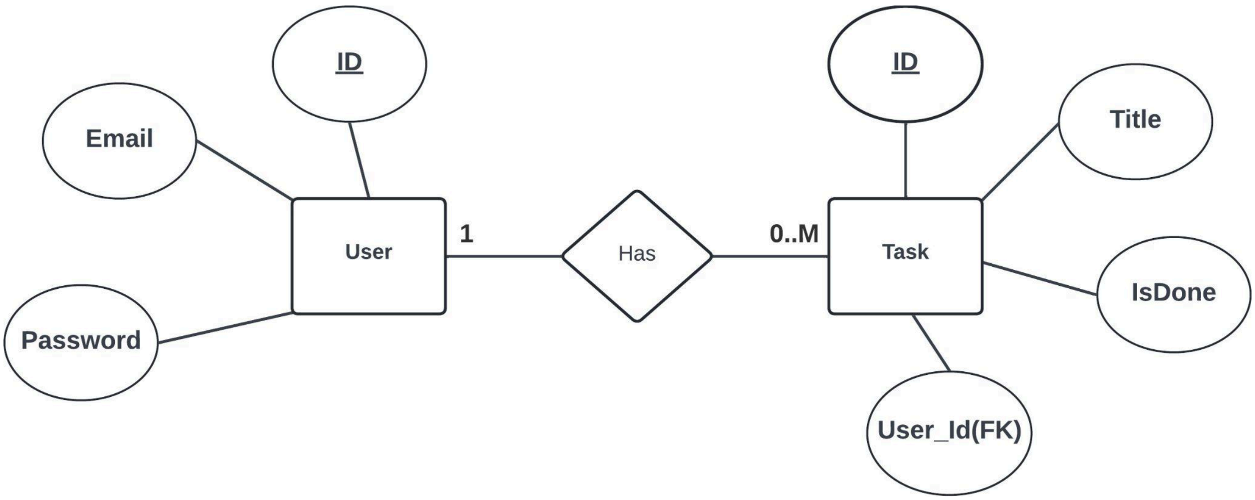
+ UpdateEmail(Id , Email)

+ UpdatePassword(Id , Password)

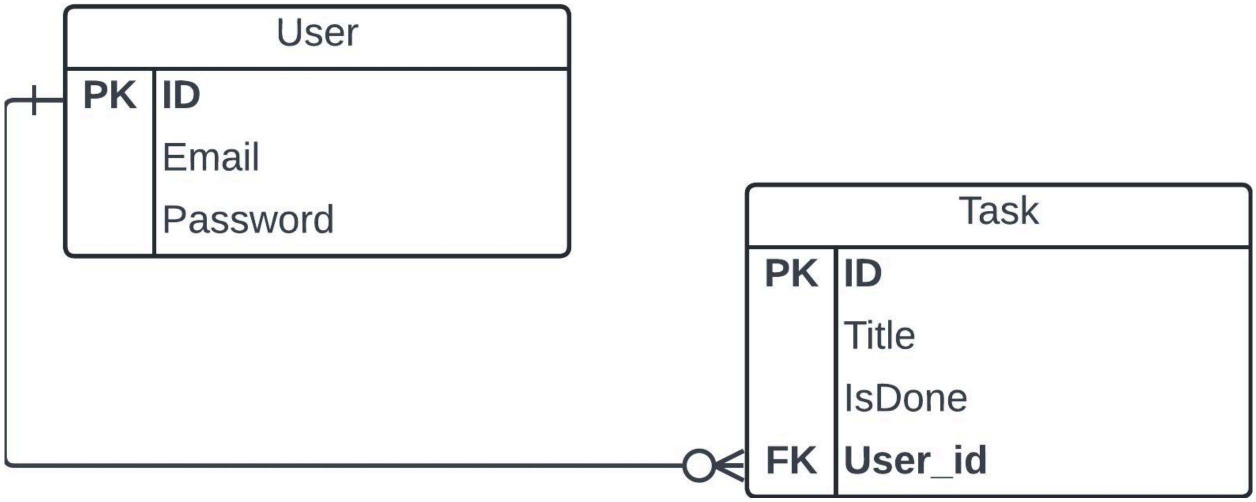
+ Set Id(Integer) : void
+ Get Id() : Integer
+ Set Email(String) : void
+ Get Email() : String
+ Set Password(String) : void
+ Get Password() : String


User
- Email:string
- Password:string
- Id:Integer
+ Signup(Email, Password)
+ Login(Email, Password)
+ UpdateEmail(Id , Email)
+ UpdatePassword(Id , Password)
+ Set Id(Integer) : void
+ Get Id() : Integer
+ Set Email(String) : void
+ Get Email() : String
+ Set Password(String) : void
+ Get Password() : String
<< Pre Condition >> p :Password <> null
<< Pre Condition >> e : Email <> null
<< Post Condition >> User.email = new email (e)
<< Post Condition >> User.password = new password (p)
<< Pre Condition >> e : Email <> null
<< Pre Condition >> p :Password <> null
<< Post Condition >> User.email = email (e)
<< Post Condition >> User.password = password (p)
<< Pre Condition >> i :id <> null
<< Pre Condition >> e : Email <> null
<< Post Condition >> User.email = new email (e)
<< Pre Condition >> i :id <> null
<< Pre Condition >> p :Password <> null
<< Post Condition >> User.password = new password (p)

ERD

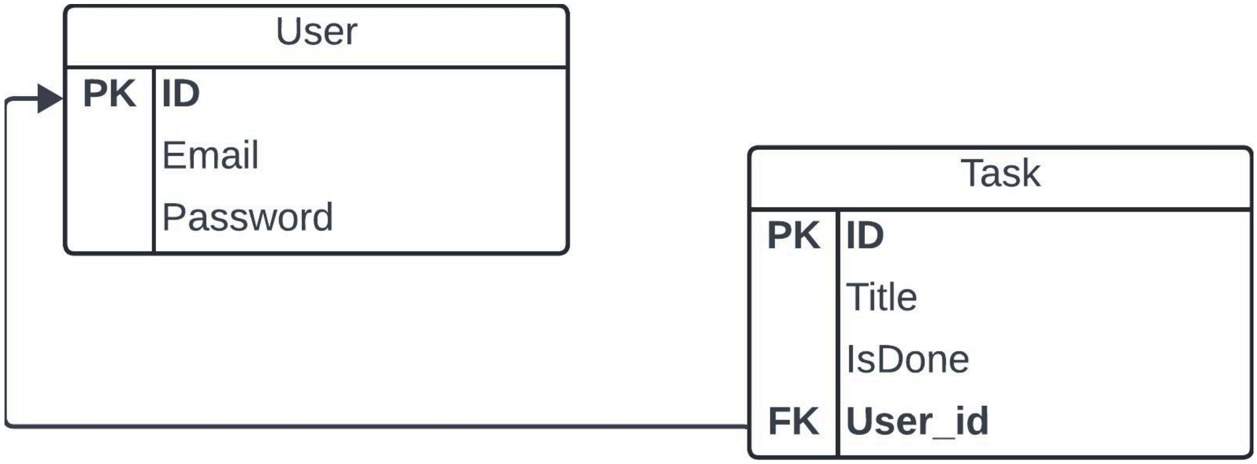


Entity Relational Schema(DB Schema)



User	
PK	ID
	Email
	Password

Task	
PK	ID
	Title
	IsDone
FK	User_id


User
PK
ID
Email
Password
Task
PK
ID
Title
IsDone
FK
User_id