Name	ID
عمر سيد سلامه	201900511
مصطفي احمد شويحي	201900809
محمد عصام محمد منشاوي	201900706
محمد عصام الدين عمر	201900705
محمود احمد محمود احمد	201900757

Print Management System

1) Introducation:-

Printer Software creates printing efficiency which in turn reduce cost and increase productivity. Print management, print monitor and adjustable security settings to prevent waste and unauthorized use are key features of our software. We have provided our solution to thousands of businesses and educational institutions. Our solutions have in turn Improved work control, reduced costs and increased effciency. Proposed system is accessed by three entities namely, Admin, User and Release Station. Admin need to login with their valid login credentials first in order to access the web application. After successful login, admin can access all the modules and perform/manage ench task accurately. Admin can perform task such as adding new user and creating login credentials for them. Admin can view or delete a registered user. System allows admin to view printing job list and approve or disapprove printing job. User can login Into the system and perform task such as uplond document for printing, and send printing request, view printing jobs, printing status and recharge their account wallet. A person at release station prints tihe desired requested document. Application handles load balancing on every

printer on the network.

Modules and their Description: The system comprises of 3 major modules with their sab-modules as follows:

1. User • Login: User need to login using valid Id and password in order to access the system. • Web Print: Upload Document (No. of pages, copies, and caleulate total number of pages) and add printing job. • Jobs Pending Release: User can view all users' uploaded documents as well as number of pages, documents and timing It may take. • Printer status: As per load balancing mechanism, printing job is divided between printers. • Recharge: User can recharge his account wallet.

2. Admin • Login: Admin need to login using valid login credentials. • Add User: Add user details with his personal details and can sent user Id and password to his email Id • View User: Can view registered user details. • Delete User: Can delete registered user if needed. • Print approve: he can approve for printing based on this timing will be changed automatically and load balancing. Print amount will be caleulated and deducted from student's wallet.

3. Release Station Print/Delete Document:User can login with their eredentials and print the document or delete It. Advantages of project • Easy To Install and Use • Printer Manager & Print Monitor • Print Counting & Print Tracking • Optimizing Printing workflow • Load Balancing • Authenticate Users • Enables secure printing

---

2)system contain three user :-

1) User:-

User can change his name and e-mail….can else get credit and upload document…… add document in his request……can else create a request….and canknow the state of the request..... and know some information about his request……print the document..... know the money in his account.

2) Admin:-

Admin can change his name and e-mail….can else get credit and upload document…… add document in his request……can else create a request….and canknow the state of the request..... and know some information about his request……print the document…..know the money in his account….can else add user….delete user….search about user….can print all user that have account in the system….update his password……assign task to user….approve and disapprove printing request.

---

3) Descripation of run:-

the first screen is show two field id and password to all user ,all user have id , password and controller take the two field and search in date base and send touser the second screen .

if the user :

-Admin he show six buttom (add user, upload document, add document, printthe document, delete user, search about user)

-User he show three buttom (get credit, create a request, know the money)

INTRODUCTION

PURPOSE:

Create online printing to easy printing

SCOPE:

Online printer.

DEFINITIONS:

User input: input from gui.
System input: input from database or other function's output.

OVERVIEW:

This web site helps in online printing:-

OVERALL VIEW

PRODUCT PERSPECTIVE:

1- Reduce the cost.
2- Reduce time
3- .3- Reduce overcrowding in printer place.
4- Help the users to prints their pepers easy.

USER CHARACTERISTICS:

1 As an Admin, I need to login to the system to be able to:

1.1 - View my profile. So that, I can change password or edit profile.
1.2 Manage persons as edit, delete them,add.
1.3 - View person profile to be able to get information about them.
1.4 - Logout from the system. So that, my account will be secure.
1.5 -approve registrations request
1.6 -approve printing request
1.7 -use any thing user uses it

1 As an user I need to login to the system to be able to:

1.1 - View my profile. So that, I can change password or edit profile

1.2 -make request

1.3 -print the printing request
1.4 - Logout from the system. So that, my account will be secure.

CONSTRAINTS:

1- System constraints.

SPECIFIC REQUIREMENTS

FUNCTIONAL REQUIREMENTS:

USER REQUIREMENTS

The software has two modules, Admin, Author.

ADMIN

- Can login to his personal account using the ID and password.
- Can view their own profile.
- Can change their current password with new one whenever required.
- Can update their own profile (like name , email , credit number).
- Can delete an person if required.
- Can update person info.
- Can view person info.
- Can uses the printer for any perpose
- Can logout.

user

- Can login their personal account using id and password.
- Can view their own profile.
- Can update their profile info
- Can upload their own document.
- Can change their current password with new one whenever required.
- Can make request
- Can view any info about their request
- Can print documents
- Can logout.

System requirements

ADMIN

- Can login to his personal account.
  User input: id and password.
  Output: Open home page.
  - **Situation:** Show error message in case of wrong id / password.
- Can view their own profile.
  System input: admin mail.
  Output: View profile.
- Can change their current password with new one whenever required.
  System input: admin id.
  User input: current and new password.
  Output: Show message which prompts that password changed successfully.
  - **Situation:** Show error message if the new password is the same of current password. So, data will not be changed.
  - **Situation:** Show error message if the new password pattern is wrong. So, data will not be changed.
  - **Situation:** Show error message if the current password not matching with their password.So, data will not be changed.
- Can update their own profile (password, email, and name , and credit).
  System input: admin id.
  User input: password and new data to change.
  Output: Show message which prompts that [data] updated successfully.
  - **Situation:** Show error message if the new data's pattern not matching the current pattern.
  So, data will not be changed.
  - **Situation:** Show error message if the current password not matching with their password.So, data will not be changed.
- Can delete an person if required.
  System input: admin e-mail.
  User input: admin password and user e-mail.
  Output: Show message which prompts that author deleted successfully.
  - Situation: Show error message if the current admin password not matching with their password. So, author will not be deleted.
- Can update author info (name, email, credit, password).
  System input: admin e-mail.
  User input: admin password and new data to change.
  Output: Show message which prompts that [data] updated successfully.
  - **Situation**: Show error message if the new data's pattern not matching the current pattern.
  So, data will not be changed.
  - **Situation**: Show error message if the current admin password not matching with their password. So, data will not be changed.
- Can view person info.
  User input: person e-mail.

Output: view person profile.

- **Situation**: Show error message if the author not found.

✧ Can uses the printer.

User input: admin e-mail.

Output: the admin need.

- **Situation**: Show error message if this is not admin.

✧ Can logout.

Output: open login page.

User

- Can login their personal account using e-mail and password.
  User input: id and password.
  Output: Open home page.
  - **Situation**: Show error message in case of wrong e-mail/ password.
- Can view their own profile.
  System input: author mail **Output**: View profile.
- Can change their current password with new one whenever required.
  System input: user e-mail.
  User input: current and new password.
  Output: Show message which prompts that password changed successfully.
  - **Situation**: Show error message if the new password is the same of current password. So,data will not be changed.
  - **Situation**: Show error message if the new password pattern is wrong. So, data will not bechanged.
  - **Situation**: Show error message if the current password not matching with their password.So, data will not be changed.
- Can edit/ update their details (name, email, password, credit).
  System input: user e-mail.
  User input: password and new data to change.
  Output: Show message which prompts that [data] updated successfully.
  - **Situation**: Show error message if the new data's pattern not matching the current pattern.So, data will not be changed.
  - **Situation**: Show error message if the current password not matching with their password.So, data will not be changed.
- Can upload document .
  User input: document.
  Output: Show message which prompts that [document] uploaded successfully.
  - **Situation**: Show error message if the file format not accepted.
- Can make request document .
  User input: document.
  Output: Show message which prompts that [document] uploaded successfully and show request number.
  - **Situation**: Show error message if the file format not accepted.
- Can view request info .
  User input: request num and id.
  Output: Show message which prompts that [document and request number and state and date] .
  - **Situation**: Show error message if the file format not accepted.
- Can logout.
  Output: open login page.

NON-FUNCTIONAL REQUIREMENTS:

LOOK AND FEEL REQUIREMENTS

- This system is user friendly.
- Approachable, so that people do not hesitate to use it.
- Authoritative, so that users feel they can rely on it and trust it.
- Attractive to admin or some other specific group.
- Innovative and appearing to be state of the art.
- Cool.

USABILITY AND HUMANITY REQUIREMENTS

- Centralized maintain all information.
- Easy to print the new document.
- Facilitate communication between admin and users.
- Easy to use on the first attempt by someone without training.
- The system will save significant amount of time and effort.

PERFORMANCE REQUIREMENTS

TIME PERFORMANCE:

- The web page load time should not exceed 1 second.
- Login process should not exceed 0.25 second.
- Any operation that will be done by any system user will not exceed 1 second.

ACCURACY PERFORMANCE:

- Accurate results for the search.

OPERATIONAL AND ENVIRONMENTAL REQUIREMENTS

- Based on the person information.

MAINTAINABILITY AND SUPPORT REQUIREMENTS

- The system is a web application. So, it can be opened in any browser.
- The system shall be able to be modified to cope with minor changes to user requests union rules.

CULTURAL REQUIREMENTS

- The language used in the interface should be formal and polite which includes Political Requirements.

SECURITY REQUIREMENTS

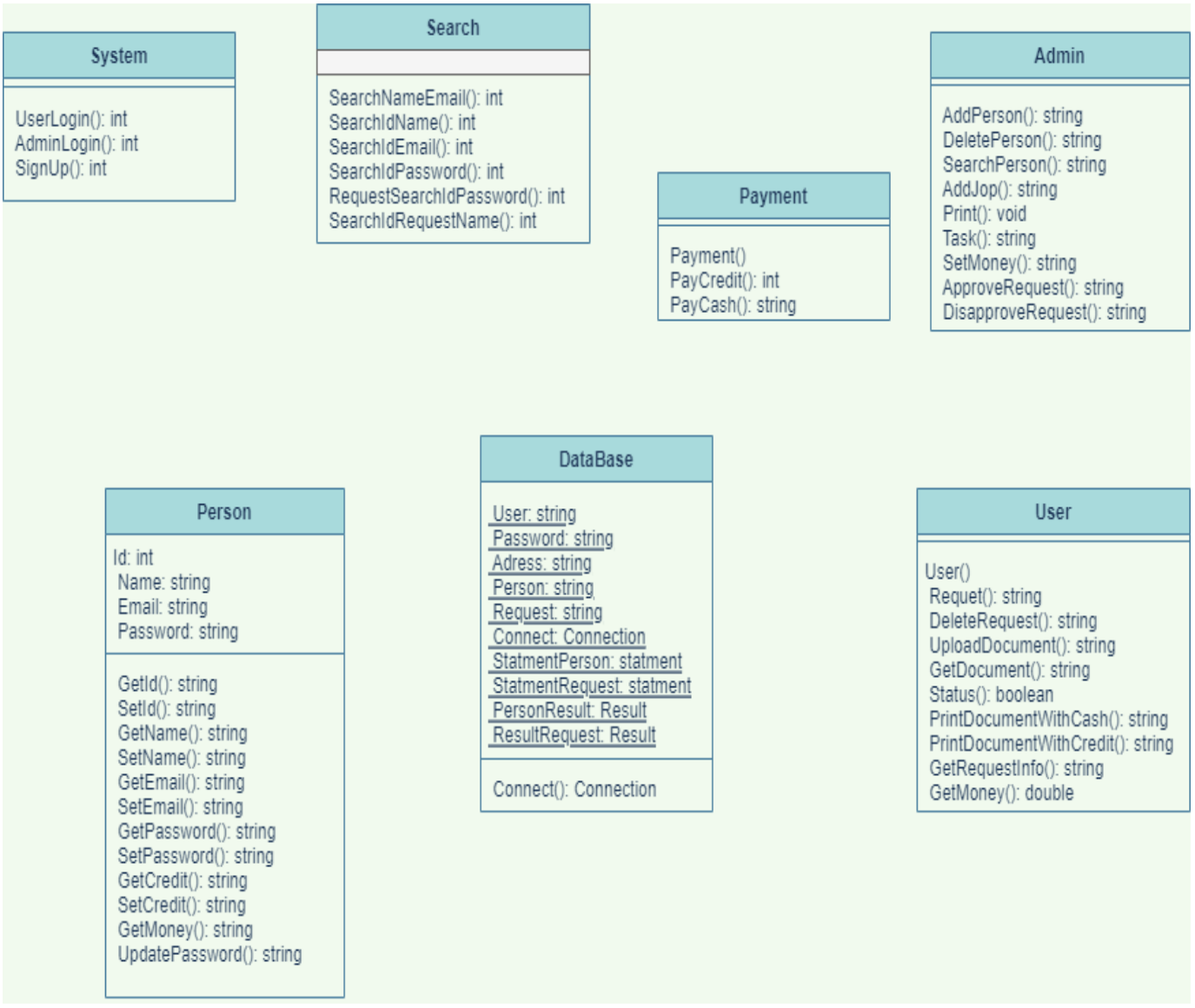
- Each modification operation requires author password.
- To use the system, it is a must to have an account and be logged in / registered.
- The system must distinguish between admin, users roles.
- Each role has his own functions which can perform. So that, no roles can be overlapped.
- Password encryption to enhance privacy and security.
- Show fake password in password field while login process instead of the one inputted by user.
- There is no way to use fake accounts. As it is required to check email to get the personal login ID.

DESIGN CONSTRAINTS

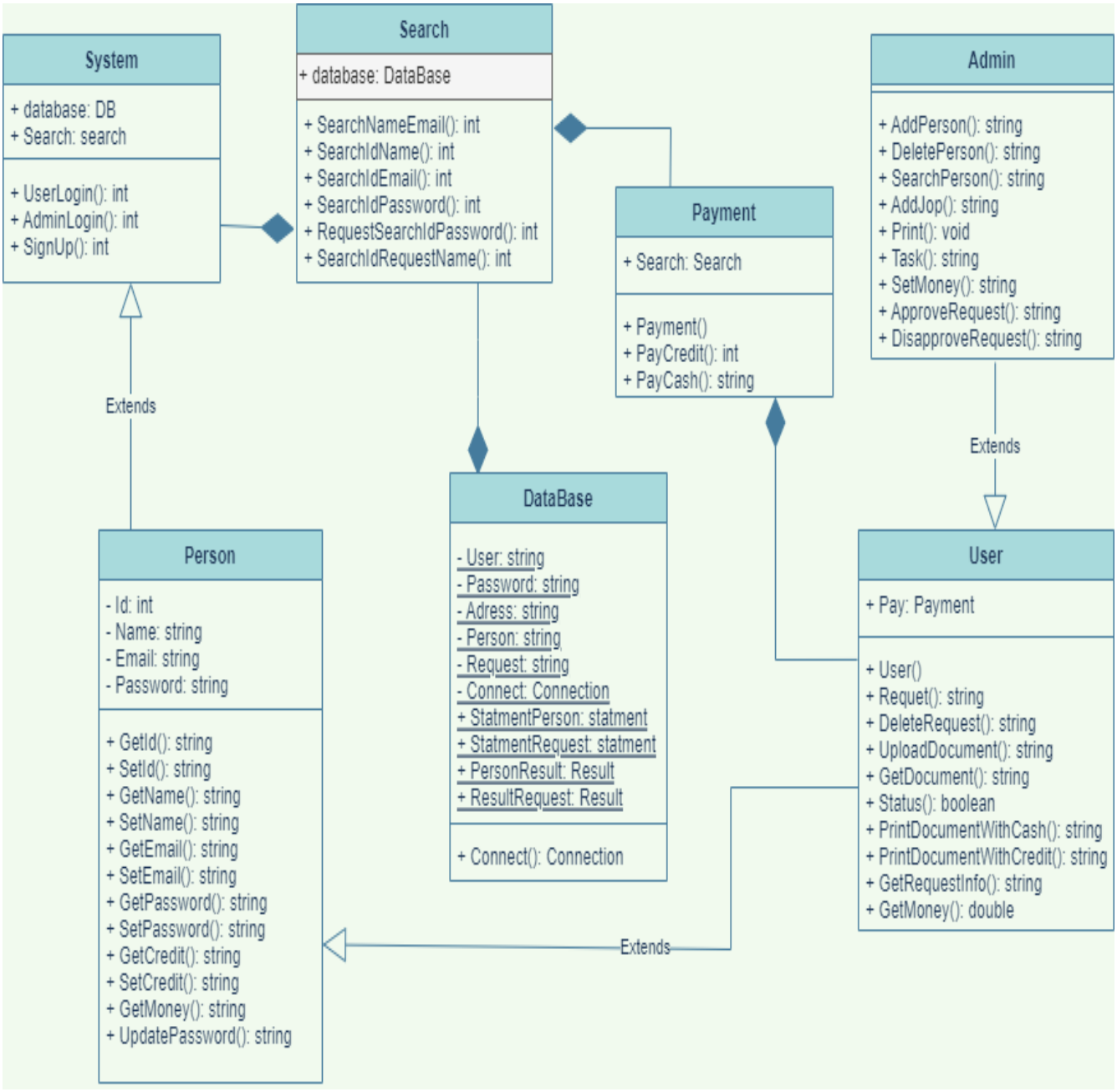
- Each password must has a pattern to follow (At least 8 characters, a mixture of both uppercase and lowercase letters, a mixture of letters and numbers, inclusion of at least one special character and don'thave <, >, ‘,”).
- inputted information like email, phone and date of birth must follow their pattern.
- Each class must have only one instance.

Class Diagram :

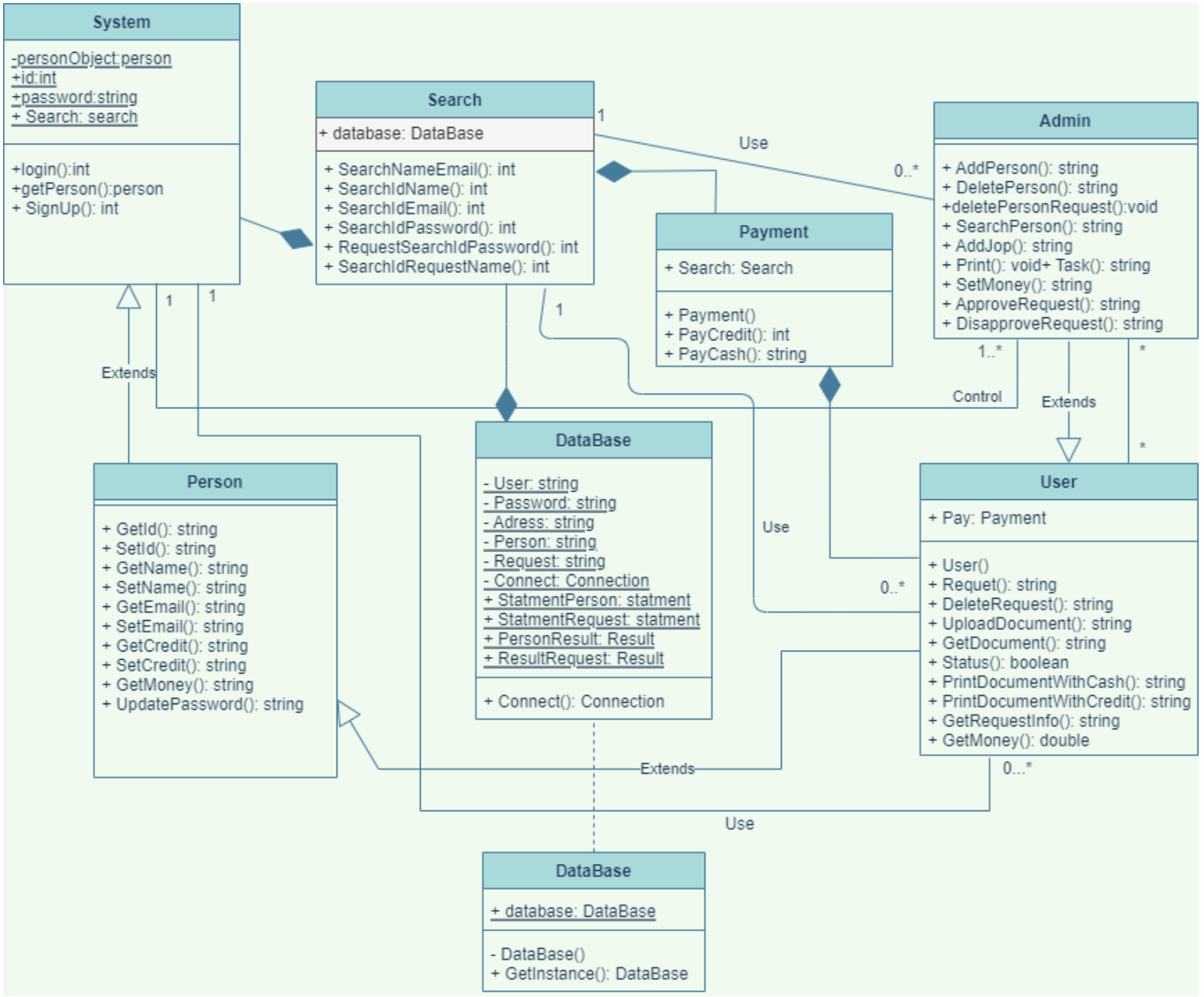
Version1:



Version 2:


System
+ database: DB
+ Search: search
+ UserLogin(): int
+ AdminLogin(): int
+ SignUp(): int
Search
+ database: DataBase
+ SearchNameEmail(): int
+ SearchIdName(): int
+ SearchIdEmail(): int
+ SearchIdPassword(): int
+ RequestSearchIdPassword(): int
+ SearchIdRequestName(): int
Payment
+ Search: Search
+ Payment()
+ PayCredit(): int
+ PayCash(): string
Admin
+ AddPerson(): string
+ DeletePerson(): string
+ SearchPerson(): string
+ AddJop(): string
+ Print(): void
+ Task(): string
+ SetMoney(): string
+ ApproveRequest(): string
+ DisapproveRequest(): string
Extends
Extends
DataBase
- User: string
- Password: string
- Adress: string
- Person: string
- Request: string
- Connect: Connection
+ StatmentPerson: statment
+ StatmentRequest: statment
+ PersonResult: Result
+ ResultRequest: Result
+ Connect(): Connection
Person
- Id: int
- Name: string
- Email: string
- Password: string
+ GetId(): string
+ SetId(): string
+ GetName(): string
+ SetName(): string
+ GetEmail(): string
+ SetEmail(): string
+ GetPassword(): string
+ SetPassword(): string
+ GetCredit(): string
+ SetCredit(): string
+ GetMoney(): string
+ UpdatePassword(): string
User
+ Pay: Payment
+ User()
+ Requet(): string
+ DeleteRequest(): string
+ UploadDocument(): string
+ GetDocument(): string
+ Status(): boolean
+ PrintDocumentWithCash(): string
+ PrintDocumentWithCredit(): string
+ GetRequestInfo(): string
+ GetMoney(): double
Extends

Version 3:


System
-personObject:person
+id:int
+password:string
+ Search: search
+login():int
+getPerson():person
+ SignUp(): int
Search
+ database: DataBase
+ SearchNameEmail(): int
+ SearchIdName(): int
+ SearchIdEmail(): int
+ SearchIdPassword(): int
+ RequestSearchIdPassword(): int
+ SearchIdRequestName(): int
Payment
+ Search: Search
+ Payment()
+ PayCredit(): int
+ PayCash(): string
Admin
+ AddPerson(): string
+ DeletePerson(): string
+deletePersonRequest():void
+ SearchPerson(): string
+ AddJop(): string
+ Print(): void+ Task(): string
+ SetMoney(): string
+ ApproveRequest(): string
+ DisapproveRequest(): string
Use
1
0..*
1
1
1
Extends
Control
1..*
*
Extends
*
Person
+ GetId(): string
+ SetId(): string
+ GetName(): string
+ SetName(): string
+ GetEmail(): string
+ SetEmail(): string
+ GetCredit(): string
+ SetCredit(): string
+ GetMoney(): string
+ UpdatePassword(): string
DataBase
- User: string
- Password: string
- Adress: string
- Person: string
- Request: string
- Connect: Connection
+ StatmentPerson: statment
+ StatmentRequest: statment
+ PersonResult: Result
+ ResultRequest: Result
+ Connect(): Connection
Use
0..*
User
+ Pay: Payment
+ User()
+ Requet(): string
+ DeleteRequest(): string
+ UploadDocument(): string
+ GetDocument(): string
+ Status(): boolean
+ PrintDocumentWithCash(): string
+ PrintDocumentWithCredit(): string
+ GetRequestInfo(): string
+ GetMoney(): double
Extends
0...*
Use
DataBase
+ database: DataBase
- DataBase()
+ GetInstance(): DataBase

Single Responsibility Principle (SRP)

Single Responsibility Principle (SRP): - is one of five design principles of the SOLID design framework for object-oriented software design. The SRP dictates that classes should have only a single reason to change

The Benefits: -

- The class is more reusable
- The class is easier to maintain
- When the class only does “one thing”, its interface The class is easier to understand usually has a small number of methods that are fairly self-explanatory.

This in classes : any function or any class don’t do more than one thing

---

Open Closed Principle (OCP)

Open Closed Principle (OCP): - In object-oriented programming, the open–closed principle states "software entities (classes, modules, functions, etc.) should be open for extension, but closed for modification"; that is, such an entity can allow its behavior to be extended without modifying its source code.

The Benefits: -

The Open Close Principle encourages software developers to design and write code in a fashion that adding new functionality would involve minimal changes to existing code. Most changes will be handled as new methods and new classes.

The difference between SRP and OCP SRP states that a class must have only one reason to change. OCP states that the class must be closed for modification but open to extension.

This in classes : like search class or payment class

Liskov Substitution Principle (LSP)

Liskov Substitution Principle (LSP): - oriented design principle that puts some restrictions on the classes that inherit other classes or implement some interfaces. It is one of the five SOLID principles that aim to make the code easier to maintain and extend in the future.

The Benefits: -

If your code adheres to the Liskov Substitution Principle you have many benefits. These include: code re-usability, reduced coupling, and easier maintenance.

This in classes : we can create object from super class using constructor of sub class

interface segregation principle (ISP)

interface segregation principle (ISP): - states that no code should be forced to depend on methods it does not use. ISP splits interfaces that are very large into smaller and more specific ones so that clients will only have to know about the methods that are of interest to them.

The Benefits: -

The Interface Segregation Principle increases the readability and maintainability of our code. We are reducing our class implementation only to required actions without any additional or unnecessary code.

This in classes : user didn’t use any operations he don’t use them like admin operations

Dependency Inversion Principle (DIP)

Dependency Inversion Principle (DIP): - In object-oriented design, the dependency inversion principle is a specific methodology for loosely coupling software modules. When following this principle, the conventional dependency relationships established from high-level, policy-setting modules to low-level, dependency modules are reversed, thusrendering high-level modules independent of the low-level module implementation details.

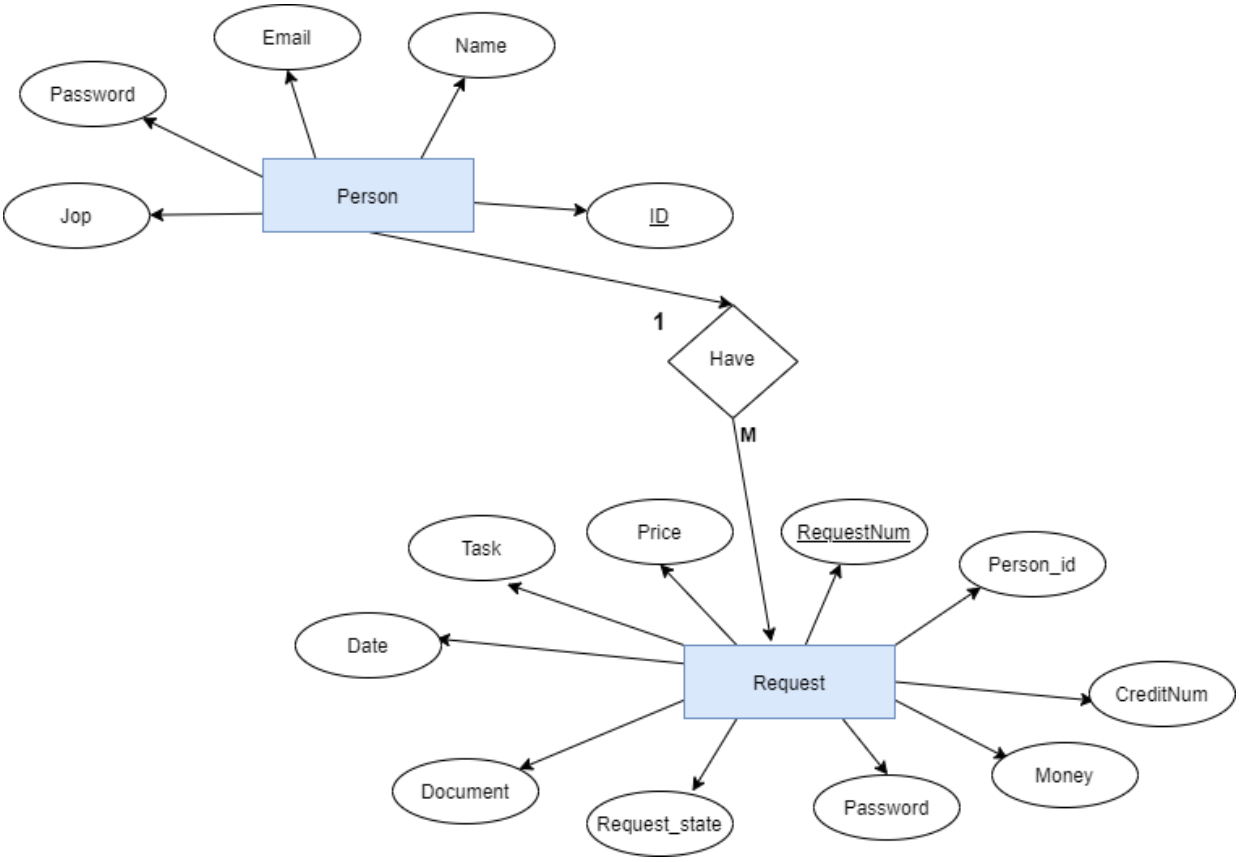
The Benefits: -

Dependency inversion well applied gives flexibility and stability at the level of the entire architecture of your application. It will allow your application to evolve more securely and stable.

This in class : no any class depends on any class from level under them

4) date base in system:-

-ERD:-



-Relational Data Model: