

ĐẠI HỌC CÔNG NGHỆ THÔNG TIN ĐHQG-TPHCM KHOA CÔNG NGHỆ THÔNG TIN

-----🙚🙘🕮🙚🙘-----

Báo cáo Công nghệ Web và ứng dụng

Mã môn: SE347.O11.PMCL

QUÁN LÝ VÀ BÁN VÉ TÀU HỎA

Giáo viên hướng dẫn:

Trần Anh Dũng

Students:

20521404 – Nguyễn Quốc Huy 20521419 – Vũ Quang Huy

19520646 – Võ Hoàng Đức Khoa

TPHCM, 12 - 2023

# Comment

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

# Document change record sheet

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Versio n | Description | Author s |
| 25/9/2021 | 1.0 | Description, system construction, functions and Initialization | All |
| 30/9/2021 | 1.1 | Analyze requirements, build Use case diagrams and design  databases | All |
| 10/10/202  1 | 1.2 | Conduct initial design of the application interface | All |
| 20/10/202  1 | 1.3 | Do the coding parallel to the interface design | All |
| 10/12/202  1 | 1.4 | Complete the functions set out as required | All |
| 15/12/202  1 | 1.5 | Handling errors and adding administrative functions for  manager | All |
| 25/12/202  1 | 1.6 | Check the software and fix the errors that arise | All |
| 30/12/202  1 | 1.7 | Make a report for the system | All |
| 5/1/2022 | 2.0 | Complete and release the system | All |

Contents

[Comment 2](#_gjdgxs)

[Document change record sheet 3](#_30j0zll)

Content 4

1. Tổng quan về đề tài [5](#_1fob9te)
   1. Mô tả bài toán [5](#_3znysh7)
   2. Mục tiêu
   3. Phạm vi
   4. Đóng góp [5](#_2et92p0)
2. Phân tích và thiết kế [5](#_tyjcwt)
   1. [Purpose of the system 5](#_3dy6vkm)
   2. [Description of users/forms and related entities 6](#_1t3h5sf)
   3. [Description of processes 9](#_4d34og8)
   4. [Functional requirements 11](#_2s8eyo1)
   5. [Non-functional requirements 13](#_17dp8vu)
   6. [Evolution requirement 13](#_3rdcrjn)
   7. [Effectiveness requirement 13](#_26in1rg)
   8. [Convenience requirements 13](#_lnxbz9)
   9. [Compatible requirements 14](#_35nkun2)
   10. [Security requirements 15](#_1ksv4uv)
   11. [Safety requirements 16](#_44sinio)
   12. [Technology requirements 16](#_2jxsxqh)
3. Công nghệ sử dụng và kết quả sản phẩm [18](#_z337ya)
   1. Công nghệ sử dụng [18](#_3j2qqm3)
      1. ReactJS
      2. NextJS
      3. TailwindCSS
      4. Typescript
      5. NodeJS
      6. MongoDB

[20](#_1ci93xb)

* 1. Demo sản phẩm
  2. Kết quả [21](#_3whwml4)
  3. [Use-case list 21](#_2bn6wsx)
  4. [23](#_qsh70q)
     1. [Use-case “Review contract” 23](#_3as4poj)
     2. [Use-case “Find contract” 24](#_1pxezwc)
     3. [Use-case “Liquidate contract” 25](#_49x2ik5)
     4. [Use-case “Send contract to customer via mail” 25](#_2p2csry)
     5. [Use-case “Create disburse certificate” 26](#_147n2zr)
     6. [Use-case “Create payment receipt” 27](#_3o7alnk)
     7. [Use-case “Create loan contract” 28](#_23ckvvd)
     8. [Use-case “Create Loan Profile” 28](#_ihv636)
     9. [Use-case “Review Loan Profile” 29](#_32hioqz)
     10. [Use-case “Find Loan Profile” 30](#_1hmsyys)
     11. [Use-case “Reject Loan Profile” 31](#_41mghml)
     12. [Use-case “Approve Loan Profile” 31](#_2grqrue)
     13. [Use-case “Delete Loan Profile” 32](#_vx1227)
     14. [Use-case “Create Loan Application” 32](#_3fwokq0)
     15. [Use-case “Review Loan Application” 33](#_1v1yuxt)
     16. [Use-case “Find Loan Application” 34](#_4f1mdlm)
     17. [Use-case “Reject Loan Application” 34](#_2u6wntf)
     18. [Use-case “Approve Loan Application” 35](#_19c6y18)
     19. [Use-case “Create customer” 36](#_3tbugp1)
  5. [Use-case “Edit Customer Information” 36](#_28h4qwu)
  6. [Use-case “Review customer” 37](#_nmf14n)
  7. [Use-case “Find customer” 38](#_37m2jsg)

1. Kết luận và hướng phát triển [39](#_1mrcu09)
   1. [Class diagram (conceptual level) 39](#_46r0co2)
   2. [Class diagram (analysis level) 40](#_2lwamvv)
   3. [Class diagram explanation 42](#_111kx3o)
      1. [Loan Profile Table: 42](#_3l18frh)
      2. [Staff Table: 44](#_206ipza)
      3. [Branch Info Table: 46](#_4k668n3)
      4. [TimeKeeping Table: 47](#_2zbgiuw)
      5. [Customer Table: 47](#_1egqt2p)
      6. [Proof Of Income Img Table: 49](#_3ygebqi)
      7. [Loan Contract Table: 50](#_2dlolyb)
      8. [Disburse Certificate Table: 51](#_sqyw64)
      9. [Liquidation Application Table: 52](#_3cqmetx)
      10. [Liquidation Decision Table: 53](#_1rvwp1q)
      11. [Payment Receipt Table: 54](#_4bvk7pj)
      12. [Exemption Application Table: 55](#_2r0uhxc)
      13. [Exemption Decision Table: 56](#_1664s55)
      14. [Extension Application Table: 58](#_3q5sasy)
      15. [Extension Decision Table: 59](#_25b2l0r)
      16. [Enum Table: 60](#_kgcv8k)
2. [Activity diagram 64](#_34g0dwd)
3. [Sign in diagram 64](#_1jlao46)
4. [Process loan profile diagram 65](#_43ky6rz)
5. [State diagram 66](#_2iq8gzs)
   1. [Authentication status diagram 66](#_xvir7l)
   2. [Loan application status diagram 67](#_3hv69ve)
   3. [Loan profile status diagram 68](#_1x0gk37)
6. [Sequence diagram 69](#_4h042r0)
   1. [Create loan profile diagram 69](#_2w5ecyt)
   2. [Create loan application diagram 70](#_1baon6m)
   3. [Create loan contract diagram 71](#_3vac5uf)
   4. [Create loan decision diagram 72](#_2afmg28)
7. [Architecture design 72](#_pkwqa1)
   1. [Application Architecture Overview 72](#_39kk8xu)
   2. [Application Architecture in detail 74](#_1opuj5n)
      1. [Front-end layer 74](#_48pi1tg)
      2. [Back-end layer 75](#_2nusc19)
8. [Interface design 76](#_1302m92)
   1. [Welcome screen 76](#_3mzq4wv)
   2. [Sign in brach screen 77](#_2250f4o)
   3. [Sign in staff screen 78](#_haapch)
   4. [Clock in/out screen (start shift) 79](#_319y80a)
   5. [Clock in/out screen (end shift) 80](#_1gf8i83)
   6. [Clock in/out screen (Summarize) 81](#_40ew0vw)
   7. [Contract screen 82](#_2fk6b3p)
   8. [Contract detail 83](#_upglbi)
   9. [Contract detail (below) 85](#_3ep43zb)
   10. [Extension decision detail screen 86](#_1tuee74)
   11. [Exemption decision detail screen 87](#_4du1wux)
   12. [Liquidation decision detail screen 88](#_2szc72q)
   13. [Add disburse screen 89](#_184mhaj)
   14. [Loan profile screen 90](#_3s49zyc)
   15. [Profile detail screen 92](#_279ka65)
   16. [Loan profile (Reject/Approve) 93](#_meukdy)
   17. [Loan profile (Create contract) 94](#_36ei31r)
   18. [Create contract 95](#_1ljsd9k)
   19. [Application screen 96](#_45jfvxd)
   20. [Application detail 97](#_2koq656)
   21. [Application detail screen (Reject/Approve) 98](#_zu0gcz)
   22. [Customer screen 99](#_3jtnz0s)
   23. [Customer info screen 101](#_1yyy98l)
   24. [Edit Customer Info screen 102](#_4iylrwe)
   25. [Admin screen 103](#_2y3w247)
   26. [Add balance 104](#_1d96cc0)
9. [Summary 105](#_3x8tuzt)
   1. [Development Environment and Deployment Environment 105](#_2ce457m)
      1. [Development Environment 105](#_rjefff)
      2. [Deployment Environment 105](#_3bj1y38)
   2. [Achievement 105](#_1qoc8b1)

[3.](#_4anzqyu) [Future development 106](#_4anzqyu)

# Overview

## Mô tả bài toán

Dự án Quản lý Quá trình Thực hiện Khóa luận Tốt nghiệp nhằm tạo ra một hệ thống hiệu quả hỗ trợ sinh viên và người hướng dẫn trong việc quản lý và thực hiện các bước quan trọng của quá trình làm khóa luận tốt nghiệp. Dưới đây là mô tả về mục tiêu và yêu cầu cơ bản của dự án:

1. Mục tiêu của dự án: Tạo ra một hệ thống quản lý thông tin toàn diện, hỗ trợ sinh viên trong việc lập kế hoạch, theo dõi tiến độ và hoàn thành khóa luận tốt nghiệp.
2. Cung cấp giao diện dễ sử dụng, giúp người dùng dễ dàng theo dõi và điều chỉnh các công việc liên quan đến khóa luận.
3. Tối ưu hóa quy trình giao tiếp và tương tác giữa sinh viên và người hướng dẫn.
4. Cung cấp công cụ phân tích và báo cáo để đánh giá tiến độ và hiệu suất của quá trình thực hiện khóa luận.

## Mục tiêu của hệ thống

* Quản lý thông tin cá nhân: Cho phép sinh viên và người hướng dẫn cập nhật thông tin cá nhân, liên lạc và thông tin liên quan đến khóa luận.
* Lập kế hoạch và theo dõi tiến độ: Cung cấp công cụ lập kế hoạch, ghi chép tiến độ và cập nhật thông tin về các giai đoạn quan trọng của khóa luận.
* Quản lý tài liệu và tài nguyên: Cho phép tải lên, chia sẻ và lưu trữ tài liệu liên quan đến khóa luận, bao gồm cả tài liệu nghiên cứu, bản nháp và tài liệu tham khảo.
* Giao tiếp và phản hồi: Cung cấp tính năng gửi thông báo, yêu cầu phê duyệt và cơ chế phản hồi giữa sinh viên và người hướng dẫn.
* Báo cáo và đánh giá: Tạo ra các báo cáo tự động về tiến độ, đánh giá hoạt động và kết quả của khóa luận.

3. Phạm vi hệ thống

4. Đóng góp

# State of the project

## Purpose of the system

* + 1. Actual demand

Hệ thống Quản lý Khóa luận Tốt nghiệp được thiết kế để tối ưu hóa và cải thiện quá trình

quản lý trong việc hoàn thành các đồ án tốt nghiệp. Giải pháp phần mềm này nhằm cung cấp một phương tiện công nghệ để giám sát, đánh giá và báo cáo về quá trình hoàn thành khóa luận

* + 1. Development opportunities

Quản lý các quy trình cũ không hiệu quả, đòi hỏi một hệ thống cho phép quản lý hiệu quả nhất đối với cả người sử dụng và sinh viên. Từ quản lý công việc của sinh viên, thông tin học liệu, hợp đồng, báo cáo tiến độ, hồ sơ nghiên cứu, v.v., sử dụng cùng một hệ thống đồng bộ và minh bạch không chỉ tiết kiệm công sức và thời gian mà còn tăng độ chính xác... Phần mềm trong tương lai sẽ đáp ứng nhu cầu của nhiều người hướng dẫn và sinh viên khi nhu cầu quản lý và thực hiện khóa luận tốt nghiệp trở nên quan trọng và đang gia tăng.

## Description of users/forms and related entities

* + 1. Plan
* Tình trạng tổ chức: Đề xuất nắm rõ cấu trúc tổ chức của hệ thống quản lý khóa luận tốt nghiệp. Tình trạng công nghệ thông tin hiện tại:
* Đánh giá trình độ phần cứng, mức độ hỗ trợ phần mềm, và năng lực về công nghệ thông tin của sinh viên và người hướng dẫn.
* Tình trạng kinh doanh: Xác định danh sách các hoạt động kinh doanh sẽ được hỗ trợ bằng việc sử dụng hệ thống quản lý khóa luận tốt nghiệp, bao gồm các yêu cầu chức năng và không chức năng.
* Những bước này giúp tạo nên một cơ sở cụ thể, từ việc hiểu rõ cấu trúc tổ chức, đánh giá mức độ công nghệ và xác định yêu cầu kinh doanh cần thiết để triển khai hệ thống quản lý khóa luận tốt nghiệp một cách hiệu quả.
  + 1. Method

Phương pháp nghiên cứu tài liệu: Để hiểu về tình trạng hiện tại của quá trình thực hiện khóa luận tốt nghiệp, bao gồm tình trạng tổ chức, công nghệ thông tin và hoạt động kinh doanh. Phương pháp phỏng vấn: Để tìm hiểu về công nghệ thông tin hiện tại và tình hình chuyên môn. Trọng tâm là khả năng sử dụng máy tính, giao diện và các yêu cầu chức năng cần thiết cho quá trình làm khóa luận.

* + 1. Current status survey
       1. Organizational status

Bộ phận quản lý khóa luận tốt nghiệp:

Chức năng chính:

Điều hành và quản lý toàn bộ quá trình thực hiện khóa luận tốt nghiệp. Nhiệm vụ:

Xác định quy trình làm việc cụ thể cho sinh viên và giáo viên hướng dẫn.

Xác nhận và quản lý các deadlines cho việc nộp báo cáo và các giai đoạn quan trọng khác của quá trình làm khóa luận.

Xử lý yêu cầu thêm đề tài và các thay đổi trong quá trình thực hiện khóa luận từ giáo viên và sinh viên.

Hệ thống quản lý thông tin:

Chức năng chính:

Quản lý và lưu trữ thông tin của sinh viên, giáo viên, đề tài, báo cáo, và deadlines.

Nhiệm vụ:

Lưu trữ thông tin về danh sách đề tài, các yêu cầu, và trạng thái xử lý của từng yêu cầu.

Quản lý deadlines và thông tin về tiến độ của khóa luận tốt nghiệp.

Người hướng dẫn (giáo viên):

Chức năng chính:

Hỗ trợ và hướng dẫn sinh viên trong quá trình thực hiện khóa luận.

Nhiệm vụ:

Đóng vai trò quan trọng trong việc hướng dẫn và đánh giá báo cáo từ sinh viên. Gửi yêu cầu thêm đề tài hoặc thay đổi đề tài.

Sinh viên:

Chức năng chính:

Thực hiện quá trình làm khóa luận tốt nghiệp.

Nhiệm vụ:

Đăng ký đề tài, theo dõi và tuân thủ các deadlines.

Nộp báo cáo và yêu cầu thay đổi đề tài (nếu cần).

Cấu trúc này giúp tổ chức và quản lý quá trình thực hiện khóa luận tốt nghiệp một cách có hệ thống và hiệu quả, đồng thời tạo điều kiện cho việc tương tác và hỗ trợ giữa sinh viên, giáo viên và các bộ phận quản lý khác nhau.

* + - 1. Current state of informatics
         * *Hardware*

Computer system configuration

|  |  |  |
| --- | --- | --- |
|  | Management department | Staff Department |
| Processor | CPU Intel Xeon Platinum 8280 / 38.5 MB / 2.7GHz  turbo / 28 cores 56 threads | AMD Ryzen Threadripper PRO 3995WX  / Socket sWRX80 / 256MB / 4.2Ghz / 64 cores 128 threads |
| RAM | 64GB | 16GB |
| Hard Disk | 1TB (HDD) and 1TB (SSD) | 1TB (HDD) and 1TB (SSD) |

|  |  |  |
| --- | --- | --- |
| Video card | GeForce RTX 3090 | Radeon RX 6900 XT |

Internet system using Viettel optical cable with 120Mbps speed to broadcast wifi through the WAN port of Ubiquiti EdgeRouter 4 with the ability to provide 500 users at the same time.

* + - * + *Software*

|  |  |  |
| --- | --- | --- |
| Name | License | Function |
| Android | X | Operator System |
| Android Studio |  | IDE/Text Editor |
| Visual Studio Code | X |
| MongoDBCompass | X | Database Management |
| MongoDB Atlas | X |

* + - * + *People, the level of informatics*
* Cashiers can easily absorb information technology.
* Marketing staff with computer certification certificates.
* HR staff have been using bank management software well.
  + - 1. Current status of the business
         * *List of occupations corresponding to the department*
* The Customer Relations Department contacts and directly contacts customers to advise and sell products that the bank provides.
* The Risk Management Department identifies, analyzes, measures and assesses risks, thereby finding measures to control and overcome the consequences of risks for business activities in order to make optimal use of resources.
* The Property Appraisal Department appraises property prices for loan purposes at banks before granting limits to help banks avoid risks.
* The Debt Settlement Department identifies and applies debt settlement solutions.
* The Administrative Department controls administrative costs of the whole system according to the annual budget plan and within the cost norms.
* The Customer Support Department updates and stores information and data collected from customers and provides them to relevant departments for the development of products and services of the bank.
* The Payment Center is in charge of the interbank payment system with modern payment centers to ensure interbank payments in the region and nationwide.
* The Credit Administration Department identifies, analyzes, and measures the level of risk, thereby proposing measures and management methods to limit and eliminate the level of credit risk in the bank's credit activities.
* The Enterprise Customer Service Department is in charge of a company's financial transactions with an organization that provides business loans, credits, accounts designed specifically for companies and businesses.
* The Personal Customer Service Department is in charge of the services and products offered to individuals and households through a network of branches.
* The Treasury Service Management Department performs work related to treasury money operations.
* The Finance and Accounting Department collects, records, processes, and analyzes economic and financial transactions, and at the same time provides necessary information for the management of monetary activities at the bank, and other financial transactions. organization, individual.
* The General Accounting Department collects, records, and summarizes economic transactions arising from business activities of the bank, providing information for the management of the bank's monetary activities.
* The Organization and Administration Department advises leaders on human resources, completing archiving, administrative clerical, and asset management tasks for the bank.

## Description of processes

* + 1. Submit loan profile

|  |  |
| --- | --- |
| Actor’s Activities | System’s reaction |
| This use case begins when a staff click “create loan profile” | |
| Actor will fill loan profile with the information that the customer provided | -System will confirm whether the entered information is valid or not, then report an error if so. |
| Actor click the send button | -Loan profile will be saved and sent to the appraisal department |

* + 1. Create appraisal report

|  |  |
| --- | --- |
| Actor’s Activities | System’s reaction |
| This use case begins when the staff clicks the appraisal loan profile. | |
| Actor click the “create appraisal report for loan profile” button | -System will display the detailed information of the appraisal report on the screen with the customer's information already entered. |
| Actor enter detailed information for the appraisal report |  |
| Actor click send appraisal report | -System will save the appraisal report into the database and send it to the approval department |

* + 1. Loan review

|  |  |
| --- | --- |
| Actor’s Activities | System’s reaction |
| This use case begins when the executive department clicks the “review ” button. | |
| Actor will click the “review” button | -System will show the loan profile and appraisal report on the screen. |
| Actor will click the “approve” button | -System will update the loan profile’s state and send it to staff for creating a loan contract |

\*Sub-event:

|  |  |
| --- | --- |
| Actor’s Activities | System’s reaction |
| This use case begins when the executive department clicks the “reject ” button. | |
| Actor will click the “reject” button | -System will update the loan profile’s state to “Rejected” |

* + 1. Create loan contract

|  |  |
| --- | --- |
| Actor’s Activities | System’s reaction |
| This use case begins when staff clicks the “create loan contract” button | |
| Actor will click the “create loan contract” button | -System will display loan contract sample on the screen. |
| Actor will enter contract information |  |
| Actor will click the “save contract” button | -System will save loan contract into the database. |

## Functional requirements

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | Operation’s name | Form | Constraint | Note |
| 1 | Loan | Form1 | Cons1 |  |
| 2 | Receive profile | Form1.1 | Cons1.1 |  |
| 3 | Validate profile | Form1.2 | Cons1.2 |  |
| 4 | Appraise customer | Form1.3 | Cons1.3 |  |
| 5 | Review profile | Form1.4 | Cons1.4 |  |
| 6 | Create loan contract | Form1.5 | Cons1.5 |  |
| 7 | Disburse | Form1.6 | Cons1.6 |  |
| 8 | Profile management | Form2 | Cons2 |  |
| 9 | Modify profile | Form2.1 | Cons2.1 |  |
| 10 | Monitor report | Form2.2 | Cons2.2 |  |
| 11 | Notify payment due date | Form2.3 | Cons2.3 |  |
| 12 | Receive, validate application | Form2.4 | Cons2.4 |  |
| 13 | Resolve application | Form2.5 | Cons2.5 |  |
| 14 | Liquidate contract | Form2.6 | Cons2.6 |  |
| 15 | Pay, report | Form3 | Cons3 |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 16 | Pay debt | Form3.1 | Cons3.1 |  |
| 17 | Create final settlement | Form3.2 | Cons3.2 |  |
| 18 | General report | Form3.3 | Cons3.3 |  |

## Non-functional requirements

## Evolution requirement

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Operation | Parameter need to be modified | Value domain need to be changed |
| 1 | Update contract detail | * Customer’s information * Contract’s state |  |

## Effectiveness requirement

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | Operation | Process time | Storage | Note |
| 1 | Create contract | 500 contracts/time |  |  |
| 2 | Search contract | 500-1000 milliseconds |  |  |
| 3 | Save contract | Instantly |  |  |
| 4 | Search application | 500-1000 milliseconds |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 5 | Notify | 1-3 seconds |  |  |
| 6 | Review profile | Instantly |  |  |
| 7 | Create report | 1-3 seconds |  |  |

## Convinience requirements

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | Operation | Easy to study | Easy to use | Note |
| 1 | Create contract | 5 minutes instruct | The average fail rate is 1%. |  |
| 2 | Search contract | 3 minutes instruct | Unnecessarily know of the contract that needs to be found | Search accurately |
| 3 | Save contract | Doesn’t need instruction |  |  |
| 4 | Search application | 3 minutes instruct | Unnecessarily know of the application that needs to be found | Search accurately |
| 5 | Notify | Doesn’t need instruction |  |  |
| 6 | Review profile | Doesn’t need instruction |  |  |
| 7 | Create report | 10 minutes instruct | Doesn’t need many actions |  |

## Compatible requirements

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Operation | Related object | Note |
| 1 | Send notification | To all of the devices installed the application | Independent of OS version or application version |
| 2 | Export profile | To printer | Independent of printer type |
| 3 | Export report |
| 4 | Export application |
| 5 | Export contract |

## Security requirements

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| No. | Operation | System admin | Board of directors | Business department | Appraisal department | Support department |
| 1 | Decentralization | X |  |  |  |  |
| 2 | Receive profile |  |  |  |  | x |
| 3 | Validate profile |  |  | x |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 4 | Appraise customer |  |  |  | x |  |
| 5 | Review profile |  | x |  |  |  |
| 6 | Create contract |  |  | x |  | x |
| 7 | Disburse |  |  | x |  | x |
| 8 | Modify contract |  |  | x |  |  |
| 9 | Monitor report |  |  | x |  |  |
| 10 | Notify payment due date |  |  | x |  | x |
| 11 | Receive, validate application |  |  |  |  | x |
| 12 | Resolve application |  | x |  |  |  |
| 13 | Liquidate contract |  |  | x |  | x |
| 14 | Pay debt |  |  | x |  | x |
| 15 | Create final settlement |  |  | x |  |  |
| 16 | General report |  |  | x |  |  |

## Safety requirements

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Operation | Object | Note |
| 1 | Recover | Loan profile, reports, applications, certificates of payment, contracts deleted |  |
| 2 | Actual delete (Can’t recover) | Loan profile, reports, applications, certificates of payment, contracts deleted |  |
| 3 | Can’t delete | * Dossiers and documents related to customers who have a valid contract with the bank. * Profiles, documents related to a customer that currently has valid contract with the bank |  |

## Technology requirements

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Requirement | Detail describe | Note |
| 1 | Fix bugs easily | Identify bugs in an average of 15 minutes | Not affect other modules when fixing this module |
| 2 | Easy to  maintain | Fix bug in an average of 1 hour |
| 3 | Easy to  upgrade | Adding a new feature or upgrade an existing feature quickly. | With requirements |

|  |  |  |  |
| --- | --- | --- | --- |
| 4 | Reuseable | Some modules can be reused in some other projects | With requirements |
| 5 | Fast completion | Build bank operation management in an average of 4 to 6 weeks | With requirements |
| 6 | Flexible | Switch to another database within 2 days | With requirements |

# Công nghệ sử dụng và demo sản phẩm

## Công nghệ sử dụng

## 1.1. ReactJS

ReactJS, thường được gọi đơn giản là React, là một thư viện JavaScript phổ biến được dùng để xây dựng giao diện người dùng, đặc biệt là các ứng dụng một trang (single-page applications - SPA). Nó được phát triển bởi Facebook (nay là Meta) và ra mắt vào năm 2013.

### a. Lý do nên sử dụng ReactJS

1. Component-Based Architecture: React sử dụng cấu trúc dựa trên các thành phần (component), giúp phát triển ứng dụng trở nên linh hoạt và dễ tái sử dụng code.
2. Virtual DOM: React tạo ra một DOM ảo (Virtual DOM), cho phép ứng dụng cập nhật hiệu quả và nhanh chóng mà không cần tái tải toàn bộ trang.
3. Hỗ trợ Lớn từ Cộng đồng: Với cộng đồng lớn, React có nhiều tài nguyên học tập, thư viện bổ trợ và công cụ phát triển.
4. Tối ưu cho Hiệu suất: Kỹ thuật so sánh và cập nhật thông minh giúp tăng hiệu suất ứng dụng.
5. JSX: JSX là cú pháp mở rộng cho JavaScript, giúp viết code giao diện trở nên dễ dàng và rõ ràng hơn.
6. Tích hợp Linh hoạt: React có thể tích hợp với nhiều framework và thư viện khác như Redux, React Router,...

### b. Điểm mạnh

1. Tối ưu hiệu suất: Nhờ Virtual DOM, React giảm thiểu thời gian cần thiết để cập nhật giao diện người dùng.
2. Tái sử dụng Component: Khả năng tái sử dụng component giúp quá trình phát triển nhanh và hiệu quả hơn.
3. Cộng đồng lớn và Hhỗ trợ mạnh mẽ: Có rất nhiều tài liệu, hướng dẫn và nguồn lực hỗ trợ từ cộng đồng.
4. Cải tiến liên tục: React liên tục được cập nhật và cải tiến, theo kịp với xu hướng công nghệ mới.
5. Tương thích ngược: Facebook chú trọng đến việc đảm bảo tương thích ngược cho các phiên bản mới của React.

### c. Điểm yếu

1. Chỉ là Thư viện, Không phải Framework: React chỉ là một thư viện chuyên về giao diện, không phải là một framework toàn diện; do đó, cần sự hỗ trợ từ các thư viện khác.
2. Học JSX có thể Khó khăn: Việc học JSX có thể gây khó khăn cho những người mới làm quen với React.
3. Cập nhật Liên tục: Những cập nhật thường xuyên đôi khi làm cho việc theo kịp với công nghệ trở nên khó khăn.
4. Quản lý Trạng thái: Trong các ứng dụng lớn, việc quản lý trạng thái có thể trở nên phức tạp và đòi hỏi các thư viện bổ trợ như Redux.

## 1.2. NextJS

Next.js là một framework React cho phép phát triển ứng dụng web React một cách dễ dàng và có hiệu suất cao. Dưới đây là một giới thiệu về Next.js, bao gồm lý do nên sử dụng, điểm mạnh và điểm yếu:

### a. Lý do nên sử dụng Next.js

1. Server-Side Rendering (SSR) và Static Site Generation (SSG): Next.js hỗ trợ SSR và SSG một cách nhanh chóng và dễ dàng. Điều này giúp cải thiện hiệu suất và tối ưu trải nghiệm người dùng.
2. Routing tích hợp: Next.js có hệ thống routing tự động, giúp quản lý các đường dẫn URL một cách dễ dàng và tự nhiên.
3. Hot Module Replacement (HMR): HMR giúp người phát triển xem những thay đổi ngay lập tức mà không cần làm mới trình duyệt, tăng tốc độ phát triển.
4. Tích hợp dễ dàng với React: Next.js được xây dựng trên nền tảng React, giúp tích hợp với các thành phần React hiện có một cách mượt mà.
5. Tích hợp API Routes: Cung cấp API Routes để xây dựng các API một cách dễ dàng và có thể được tích hợp trực tiếp vào ứng dụng Next.js.
6. Cộng đồng và Hỗ trợ: Next.js có một cộng đồng lớn và đội ngũ phát triển đằng sau được hỗ trợ chặt chẽ.

### b. Điểm mạnh

1. Hiệu suất Cao: SSR và SSG giúp giảm thời gian tải trang, cải thiện trải nghiệm người dùng và tối ưu hóa SEO.
2. Dễ Dàng Tích Hợp với API: Có thể tích hợp dễ dàng với các API bằng cách sử dụng API Routes, làm cho việc xây dựng và quản lý ứng dụng hoạt động trơn tru và hiệu quả.
3. Hỗ Trợ CSS-in-JS: Next.js hỗ trợ nhiều cách quản lý CSS, bao gồm CSS modules, styled-components, và emotion.
4. Phát Triển Nhanh chóng: HMR giúp tăng tốc quá trình phát triển bằng cách hiển thị những

thay đổi ngay lập tức.

1. Tích hợp TypeScript dễ dàng: Hỗ trợ TypeScript nền tảng, giúp kiểm soát kiểu dữ liệu và tăng tính ổn định của ứng dụng.

### c. Điểm yếu

1. Nhiều kiến thức cần nắm trước khi vận dụng: Có một số khái niệm và tính năng mới (ví dụ: Server-Side Rendering) có thể đòi hỏi thời gian để hiểu rõ, đặc biệt là đối với những người mới bắt đầu.
2. Tích hợp cụm cơ sở dữ liệu phức tạp: Khi ứng dụng phức tạp, việc quản lý cơ sở dữ liệu có thể đòi hỏi sự xử lý cẩn thận và kiến thức sâu rộng về Next.js và React.
3. Cần hiểu rõ React: Đối với những người mới sử dụng Next.js, việc có kiến thức vững về React là quan trọng để tận dụng đầy đủ tiềm năng của framework.

## 1.3. TailwindCSS

Tailwind CSS là một framework CSS utility-first, có ý nghĩa là nó cung cấp một tập hợp các lớp CSS có sẵn để bạn có thể sử dụng trực tiếp trong HTML của mình. Dưới đây là một giới thiệu về Tailwind CSS, bao gồm lý do nên sử dụng, điểm mạnh và điểm yếu:

### a. Lý do nên sử dụng Tailwind CSS

1. Tốc độ Phát triển Nhanh chóng: Tailwind giúp tăng tốc quá trình phát triển bằng cách giảm thiểu việc viết CSS từ đầu, thay vào đó sử dụng các lớp CSS đã được định nghĩa sẵn.
2. Dễ Dàng Tùy chỉnh và Mở Rộng: Mặc dù cung cấp nhiều lớp CSS mặc định, nhưng Tailwind cũng cho phép bạn tùy chỉnh và mở rộng theo ý muốn của bạn thông qua tệp cấu hình.
3. Khả năng Tương tác và Hiệu quả cao: Tailwind giúp tối ưu hóa trải nghiệm người dùng bằng cách giảm kích thước file CSS và tối ưu hóa hiệu suất của trang web.
4. Không Cần Chọn Lựa Tên Lớp: Bạn không cần phải đặt tên cho lớp CSS của mình, giúp tránh những quyết định khó khăn về tên lớp.
5. Cộng Đồng Lớn và Hỗ Trợ Mạnh Mẽ: Tailwind có một cộng đồng rộng lớn, có nhiều tài nguyên hỗ trợ và ví dụ sử dụng.

### b. Điểm mạnh

1. Tăng tốc phát triển: Việc sử dụng các lớp đã được định nghĩa giảm thời gian viết CSS và tăng tốc độ phát triển.
2. Tùy chỉnh linh hoạt: Dù có sẵn các lớp CSS, bạn vẫn có thể tùy chỉnh và mở rộng để phù hợp với thiết kế của bạn.
3. Khả năng tích hợp tốt với các framework và thư viện khác: Tailwind dễ tích hợp với các

framework JavaScript như React, Vue.js, và Angular.

1. Tiếp cận Utility-First: Giúp tăng khả năng tái sử dụng và giảm độ phức tạp của CSS.

### c. Điểm yếu

1. Dung Lượng Lớn: Khi sử dụng đầy đủ các lớp, dung lượng của tệp CSS có thể trở nên lớn, tuy nhiên, có thể tối ưu hóa bằng cách loại bỏ các phần không cần thiết.
2. Khả năng Đọc và Hiểu Code: Một số người có thể thấy khó khăn khi đọc và hiểu code do sự tiếp cận utility-first, đặc biệt là đối với những người mới sử dụng Tailwind.
3. Tính Nhất quán Có thể Bị Mất Mát: Do sự linh hoạt và tùy chỉnh, có thể xảy ra tình trạng thiếu nhất quán giữa các phần của dự án.

## 1.4. Typescript

### a. Giới thiệu về TypeScript

TypeScript là một ngôn ngữ lập trình được phát triển bởi Microsoft, là một siêu tập (superset) của JavaScript. Nó thêm vào các tính năng của JavaScript, đặc biệt là hệ thống kiểu dữ liệu tĩnh (static typing). TypeScript được biên dịch thành JavaScript, có nghĩa là nó có thể chạy trên bất kỳ trình duyệt, máy chủ hoặc môi trường nào khác mà JavaScript có thể chạy.

### b. Lý do nên sử dụng TypeScript

1. Hệ thống kiểu dữ liệu tĩnh: TypeScript cung cấp kiểm tra kiểu tại thời điểm biên dịch, giúp phát hiện lỗi sớm và giảm thiểu lỗi tại thời điểm runtime.
2. Tính tương thích với JavaScript: Mọi mã JavaScript đều là mã TypeScript hợp lệ, điều này làm cho việc chuyển đổi từ JavaScript sang TypeScript trở nên dễ dàng.
3. Cải thiện trải nghiệm lập trình: Tính năng tự động hoàn thành mã và gợi ý kiểu dữ liệu giúp cải thiện đáng kể trải nghiệm lập trình.
4. Hỗ trợ lập trình hướng đối tượng: TypeScript hỗ trợ các tính năng OOP (Object-Oriented Programming) như classes, interfaces, và inheritance.
5. Phù hợp với dự án lớn: TypeScript thường được ưa chuộng trong các dự án lớn và phức tạp, nơi mà quản lý và bảo trì mã nguồn trở nên quan trọng.

### c. Điểm mạnh

1. Phát hiện lỗi sớm: Hệ thống kiểu giúp phát hiện lỗi trong quá trình biên dịch thay vì tại runtime.
2. Tính năng mở rộng và tính tương thích: Mở rộng JavaScript và hoàn toàn tương thích với nó.
3. Hỗ trợ IDE tốt: Hỗ trợ tốt từ các môi trường phát triển tích hợp (IDEs), với các tính năng như tự động hoàn thành mã và refactor code.
4. Dễ dàng tổ chức mã: Hỗ trợ modules, namespaces, và các tính năng hướng đối tượng giúp tổ chức và quản lý mã dễ dàng hơn.
5. Cộng đồng mạnh mẽ: Hưởng lợi từ cộng đồng lớn của JavaScript và sự hỗ trợ mạnh mẽ từ Microsoft.

### d. Điểm yếu

1. Đường học khó hơn JavaScript: Yêu cầu hiểu biết về hệ thống kiểu dữ liệu và OOP, có thể là một thách thức đối với lập trình viên mới hoặc những người chỉ quen với JavaScript.
2. Thời gian Biên Dịch: Có thể mất thêm thời gian biên dịch so với JavaScript thuần.
3. Cần Cập Nhật Kiểu Dữ Liệu cho Các Thư viện Bên Ngoài: Một số thư viện JavaScript có thể không có kiểu dữ liệu TypeScript sẵn có, yêu cầu phải tự định nghĩa hoặc tìm kiếm kiểu dữ liệu từ cộng đồng.
4. Khả Năng Tương Thích Ngược: Trong một số trường hợp, việc tích hợp TypeScript vào một dự án JavaScript hiện có có thể gây ra vấn đề về khả năng tương thích ngược.

## 1.5. NodeJS

### a. Giới thiệu về Node.js

Node.js là một môi trường chạy JavaScript trên phía máy chủ (server-side), dựa trên V8 Engine của Google Chrome. Nó được thiết kế để xây dựng các ứng dụng mạng quy mô lớn và có khả năng xử lý đồng thời nhiều kết nối một cách hiệu quả. Node.js sử dụng mô hình non-blocking, event-driven I/O, giúp nó nhẹ và hiệu quả, phù hợp với các ứng dụng thời gian thực trên phân tán dữ liệu.

### b. Lý do nên sử dụng Node.js

1. Hiệu suất cao: Node.js sử dụng V8 JavaScript Engine của Google, cho phép nó xử lý các tác vụ nhanh chóng.
2. Xử lý đồng thời: Node.js sử dụng mô hình non-blocking I/O, giúp xử lý đồng thời nhiều kết nối hiệu quả.
3. JavaScript trên máy chủ: Cho phép sử dụng JavaScript cả trên phía client lẫn server, giúp đơn giản hóa việc phát triển các ứng dụng web.
4. Hệ sinh thái lớn: npm, hệ thống quản lý gói của Node.js, có một thư viện lớn các module có sẵn, hỗ trợ đắc lực cho việc phát triển.
5. Mô hình dễ mở rộng: Phù hợp với việc xây dựng các ứng dụng web quy mô lớn và dễ dàng mở rộng.

### 

### c. Điểm mạnh

1. Khả năng xử lý đồng thời cao: Phù hợp cho các ứng dụng cần xử lý nhiều kết nối cùng lúc, như chat online, trò chơi trực tuyến.
2. Hiệu suất tốt: Nhanh chóng và hiệu quả nhờ mô hình non-blocking và sử dụng JavaScript.
3. Phát triển đơn giản và nhanh chóng: Việc sử dụng JavaScript trên cả client và server giúp đơn giản hóa quá trình phát triển.
4. Hệ sinh thái mạnh mẽ: Có sẵn hàng nghìn module qua npm, giúp dễ dàng mở rộng và tích hợp.
5. Cộng đồng lớn và sôi nổi: Hỗ trợ tốt từ cộng đồng, với nhiều tài liệu và nguồn học tập.

### d. Điểm yếu

1. Không thích hợp cho tác vụ tính toán nặng CPU: Do bản chất single-threaded, nó không phù hợp cho các ứng dụng cần nhiều xử lý CPU.
2. Callback Hell: Trong mô hình non-blocking, việc xử lý nhiều tác vụ đồng thời có thể dẫn đến "callback hell", khiến code khó đọc và quản lý.
3. Không ổn định: Một số API của Node.js vẫn đang trong giai đoạn phát triển và thường xuyên thay đổi, có thể gây khó khăn trong việc bảo trì.
4. Học curve: Đối với những người mới, việc làm quen với mô hình non-blocking và asynchronous có thể mất thời gian.
5. Quản lý bộ nhớ: Node.js có thể không hiệu quả trong việc quản lý bộ nhớ cho các ứng dụng lớn.

## 1.6. MongoDB

### a. Giới thiệu về MongoDB

MongoDB là một hệ thống quản lý cơ sở dữ liệu (Database Management System - DBMS) thuộc loại NoSQL, được thiết kế để lưu trữ và truy xuất dữ liệu một cách linh hoạt và hiệu quả. MongoDB sử dụng mô hình dữ liệu JSON-similar được gọi là BSON (Binary JSON) và không yêu cầu định cấu trúc cố định cho dữ liệu.

### b. Lý do nên sử dụng MongoDB

1. Tính linh hoạt và đa dạng: MongoDB không yêu cầu cấu trúc cố định cho dữ liệu, giúp nó phù hợp với các ứng dụng với dữ liệu đa dạng và thay đổi thường xuyên.
2. Tích hợp tốt với JavaScript: MongoDB sử dụng BSON, một định dạng tương tự JSON, phù hợp với việc làm việc với JavaScript trên cả client và server.
3. Tính tương thích với cơ sở dữ liệu lớn: MongoDB có khả năng mở rộng dễ dàng bằng cách thêm vào các node trong cụm cơ sở dữ liệu, giúp đáp ứng với nhu cầu của các

ứng dụng với lượng dữ liệu lớn.

1. Hỗ trợ truy vấn phức tạp: MongoDB hỗ trợ truy vấn phức tạp và indexing, cho phép truy xuất dữ liệu một cách nhanh chóng.
2. Tính Nhanh và Hiệu quả: Với cơ chế lưu trữ dữ liệu linh hoạt và việc sử dụng indexing, MongoDB có thể cung cấp hiệu suất cao cho việc truy xuất dữ liệu.

### c. Điểm mạnh

1. Linh hoạt với Schema: MongoDB cho phép sử dụng schema linh hoạt hoặc không cần schema, giúp nhanh chóng thích ứng với sự thay đổi trong dữ liệu.
2. Dễ Mở rộng và Phân tán: Có thể dễ dàng mở rộng cụm cơ sở dữ liệu và phân tán dữ liệu trên nhiều máy chủ.
3. Chú ý đến hiệu suất: MongoDB có thể cung cấp hiệu suất cao, đặc biệt là trong các ứng dụng với lưu lượng dữ liệu lớn và nhiều truy vấn.
4. Tích hợp tốt với JavaScript và JSON: Được tạo ra với JavaScript và JSON, MongoDB hỗ trợ sự tích hợp tốt với các ứng dụng web phía client.
5. Cộng đồng và tài liệu lớn: MongoDB có một cộng đồng sôi nổi và nhiều tài liệu học tập và hỗ trợ từ cộng đồng.

### d. Điểm yếu

1. Khả năng tương thích ngược: MongoDB không hỗ trợ join, và việc thay đổi cấu trúc dữ liệu có thể đôi khi tạo ra khả năng tương thích ngược.
2. Khả năng quản lý bộ nhớ và CPU: Dữ liệu lớn có thể tạo ra áp lực lớn cho bộ nhớ và CPU của hệ thống.
3. Truy vấn phức tạp: Trong trường hợp truy vấn phức tạp, việc sử dụng các kỹ thuật khác như MapReduce có thể đôi khi làm cho mã nguồn trở nên phức tạp.
4. Hạn chế trong việc thực hiện transactions: So với một số cơ sở dữ liệu quan hệ, MongoDB có hạn chế trong việc thực hiện giao dịch.

## Actor list

|  |  |  |
| --- | --- | --- |
| No | Name of Actor | Short description/Notes |
| 1 | Board of  directors | In charge of a bank branch, the only actor that has permission  to accept or reject loan profiles. |
| 2 | Business  department | Working with document, or support customer |
| 3 | Appraisal  department | Appraise customer and create appraisal report |
| 4 | Support  department | Mostly work with customers |
| 5 | System | Storage and handling automated works |

## Use-case list

|  |  |  |  |
| --- | --- | --- | --- |
| No | Actor | Name of Use-case | Short description/Notes |
| 1. | Support Department, Business Department, Appraisal Department,  Board of Director | Review contract |  |
| 2. | Support Department, Business Department, Appraisal Department,  Board of Director | Find contract |  |
| 3. | Support Department, Business Department, Appraisal Department,  Board of Director | Liquidate contract |  |

|  |  |  |  |
| --- | --- | --- | --- |
| 4. | System | Send contract to customer via mail |  |
| 5. | Support Department, Business Department, Appraisal Department,  Board of Director | Create disburse certificate |  |
| 6. | Board of Director | Create payment receipt |  |
| 7. | Board of Director | Create Loan Contract |  |
| 8. | Support Department, Business Department, Appraisal Department,  Board of Director | Create Loan Profile |  |
| 9. | Support Department, Business Department, Appraisal Department,  Board of Director | Review Loan profile |  |
| 10. | Support Department, Business Department, Appraisal Department,  Board of Director | Find Loan Profile |  |
| 11. | Board of Director | Reject Loan Profile |  |
| 12. | Board of Director | Approve Loan Profile |  |
| 13. | Board of Director | Delete Loan Profile |  |
| 14. | Support Department, Business Department, Appraisal Department,  Board of Director | Create Loan Application |  |
| 15. | Support Department, Business Department, Appraisal Department,  Board of Director, | Review Loan Application |  |

|  |  |  |  |
| --- | --- | --- | --- |
| 16. | Support Department, Business Department, Appraisal Department,  Board of Director | Find Loan Application |  |
| 17. | Board of Director | Reject Loan Application |  |
| 18. | Board of Director | Approve Loan Application |  |
| 19. | Board of Director | Delete Loan Application |  |
| 20. | Support Department, Business Department, Appraisal Department,  Board of Director | Create Customer |  |
| 21. | Support Department, Business Department, Appraisal Department,  Board of Director | Edit Customer Info |  |
| 22. | Support Department, Business Department, Appraisal Department,  Board of Director | Review Customer |  |
| 23. | Support Department, Business Department, Appraisal Department,  Board of Director | Find Customer |  |

## Use- case specification

## Use-case “Review contract”

|  |  |
| --- | --- |
| Use case’s name | Review contract |

|  |  |
| --- | --- |
| Actor | Support Department, Business Department, Appraisal Department, Board of Director |
| Pre-condition | The actor must be authenticated |
| Basic flow | 1. Actor clicks the Contract button on the left side tab. 2. List of contract will be shown on UI. 3. Actor click on one of the contract form that list 4. Detail information about that contract will be shown |
| Alternative flow |  |
| Post-condition | The system is ready for the next operation. |
| Extend point | None |

## Use-case “Find contract”

|  |  |
| --- | --- |
| Use case’s name | Find Contract |
| Actor | Support Department, Business Department, Appraisal Department, Board of Director |
| Pre-condition | The actor must be authenticated |
| Basic flow | 1. Actor clicks the Contract button on the left side tab. 2. Contract Dashboard will be shown 3. Actor enter the information need to find (contract number, customer phone,…) 4. List of contract result will be shown on UI. |

|  |  |
| --- | --- |
| Alternative flow | 2.1) Found nothing => application will show empty state. |
| Post-condition | The system is ready for the next operation. |
| Extend point | None |

## Use-case “Liquidate contract”

|  |  |
| --- | --- |
| Use case’s name | Liquidate contract |
| Actor | Support Department, Business Department, Appraisal Department, Board of Director |
| Pre-condition | The actor must be authenticated |
| Basic flow | 1. Actor clicks the “Application” button on the left side tab. 2. Application Dashboard will be shown on UI. 3. Actor click on the “Create” button from the Dashboard. 4. A drop down will be shown 5. Actor select “Liquidation Application” menu item. 6. Actor fill the Liquidation Application information. 7. Actor click “Save” button. 8. The application will be created. |
| Alternative flow | 7.1) The information is incorrect (missing some field, invalid format) => System will show error dialog. |
| Post-condition | The system is ready for the next operation. |
| Extend point | None |

## Use-case “Send contract to customer via mail”

|  |  |
| --- | --- |
| Use case’s name | Send contract to customer via mail |
| Actor | System |
| Pre-condition | System is running smoothly, use case “Create Loan Contract” has just happened |
| Basic flow | 1. System will create a pdf file base on the loan contract has just been created. 2. System will send the pdf file to customer by their email. |
| Alternative flow |  |
| Post-condition | The system is ready for the next operation. |
| Extend point | None |

## Use-case “Create disburse certificate”

|  |  |
| --- | --- |
| Use case’s name | Create disburse certificate |
| Actor | Support Department, Business Department, Appraisal Department, Board of Director |
| Pre-condition | The actor must be authenticated |
| Basic flow | 1) Actor clicks the Contract button on the left side tab. |

|  |  |
| --- | --- |
|  | 1. List of contract will be shown on UI. 2. Actor click on one of the contract form that list. 3. Detail information about that contract will be shown. 4. Actor click button “Add” next to text “Disburse certificate”. 5. A create dialog will be shown. 6. Actor enter the amount to be disbursed. 7. Actor click save. 8. The disburse certificate will be created |
| Alternative flow |  |
| Post-condition | The system is ready for the next operation. |
| Extend point | None |

## Use-case “Create payment receipt”

|  |  |
| --- | --- |
| Use case’s name | Create payment receipt |
| Actor | Board of Director |
| Pre-condition | The actor must be authenticated |
| Basic flow | 1. Actor clicks the Contract button on the left side tab. 2. List of contract will be shown on UI. 3. Actor click on one of the contract form that list. 4. Detail information about that contract will be shown. 5. Actor click one of the liquidation applications from the list. 6. Actor click “Add payment receipt” button. 7. The payment receipt will be created |
| Alternative flow |  |

|  |  |
| --- | --- |
| Post-condition | The system is ready for the next operation. |
| Extend point | None |

## Use-case “Create loan contract”

|  |  |
| --- | --- |
| Use case’s name | Create loan contract |
| Actor | Board of Director |
| Pre-condition | * The actor must be authenticated. * The use case “Review Loan Profile” has just happened. * Current reviewing profile hasn’t had contract yet. * Current reviewing profile was approved. |
| Basic flow | 1. Actor click “Create contract button” 2. A create contract dialog will be shown. 3. Actor enter required information. 4. Actor click “Save” button. 5. Contract will be created. |
| Alternative flow | 3.1) The information is incorrect (missing some field, invalid format)  => System will show error dialog. |
| Post- condition | The system is ready for the next operation. |
| Extend point | None |

## Use-case “Create Loan Profile”

|  |  |
| --- | --- |
| Use case’s name | Create Loan Profile |
| Actor | Support Department, Business Department, Appraisal Department, Board of Director |
| Pre-condition | The actor must be authenticated. |
| Basic flow | 1. Actor clicks the “Profile” button on the left side tab. 2. Profile Dashboard will be shown on UI. 3. Actor click on the “Create” button from the Dashboard. 4. Actor select customer. 5. A Profile templete will be shown 6. Actor enter required information to create loan profile. 7. Actor click next to navigate to review information. 8. Actor click “Save” button 9. Profile will be created |
| Alternative flow | 7.1) The information is incorrect (missing some field, invalid format)  => System will show error dialog. |
| Post- condition | The system is ready for the next operation. |
| Extend point | None |

## Use-case “Review Loan Profile”

|  |  |
| --- | --- |
| Use case’s name | Review Loan Profile |
| Actor | Support Department, Business Department, Appraisal Department, Board of Director |
| Pre-condition | The actor must be authenticated |
| Basic flow | 1. Actor clicks the “Profile” button on the left side tab. 2. List of profile will be shown on UI. 3. Actor click on one of the profile form that list 4. Detail information about that profile will be shown |
| Alternative flow |  |
| Post-condition | The system is ready for the next operation. |
| Extend point | None |

## Use-case “Find Loan Profile”

|  |  |
| --- | --- |
| Use case’s name | Find Loan Profile |
| Actor | Support Department, Business Department, Appraisal Department, Board of Director |
| Pre-condition | The actor must be authenticated |
| Basic flow | 1. Actor clicks the Profile button on the left side tab. 2. Profile Dashboard will be shown. |

|  |  |
| --- | --- |
|  | 1. Actor enter the information need to find (profile number, customer phone,…) 2. List of contract result will be shown on UI. |
| Alternative flow | 2.2) Found nothing => application will show empty state. |
| Post-condition | The system is ready for the next operation. |
| Extend point | None |

## Use-case “Reject Loan Profile”

|  |  |
| --- | --- |
| Use case’s name | Reject Loan Profile |
| Actor | Board of Director |
| Pre-condition | * The actor must be authenticated. * The use case “Review Loan Profile” has just happened. * The Profile has not been approved. |
| Basic flow | 1. Actor click “Reject” button. 2. The profile will be rejected |
| Alternative flow |  |
| Post-condition | The system is ready for the next operation. |
| Extend point | None |

## Use-case “Approve Loan Profile”

|  |  |
| --- | --- |
| Use case’s name | Approve Loan Profile |
| Actor | Board of Director |
| Pre-condition | * The actor must be authenticated. * The use case “Review Loan Profile” has just happened. * The Profile has not been approved. |
| Basic flow | 1. Actor click “Approve” button. 2. The profile will be approved. |
| Alternative flow |  |
| Post-condition | The system is ready for the next operation. |
| Extend point | None |

## Use-case “Delete Loan Profile”

|  |  |
| --- | --- |
| Use case’s name | Delete Loan Profile |
| Actor | Board of Director |
| Pre-condition | * The actor must be authenticated. * The use case “Review Loan Profile” has just happened. * The Profile has not been approved. |
| Basic flow | 1. Actor click “Delete” icon button. 2. The profile will be deleted. |
| Alternative flow |  |

|  |  |
| --- | --- |
| Post-condition | The system is ready for the next operation. |
| Extend point | None |

## Use-case “Create Loan Application”

|  |  |
| --- | --- |
| Use case’s name | Create Loan Application |
| Actor | Support Department, Business Department, Appraisal Department, Board of Director |
| Pre-condition | The actor must be authenticated |
| Basic flow | 1. Actor clicks the “Application” button on the left side tab. 2. Application Dashboard will be shown on UI. 3. Actor click on the “Create” button from the Dashboard. 4. A drop down will be shown 5. Actor select application type in dropdown menu. 6. Actor fill the required information. 7. Actor click “Save” button. 8. The application will be created. |
| Alternative flow | 7.1) The information is incorrect (missing some field, invalid format) => System will show error dialog. |
| Post-condition | The system is ready for the next operation. |

|  |  |
| --- | --- |
| Extend point | None |

## Use-case “Review Loan Application”

|  |  |
| --- | --- |
| Use case’s name | Review Loan Application |
| Actor | Support Department, Business Department, Appraisal Department, Board of Director |
| Pre-condition | The actor must be authenticated |
| Basic flow | 1. Actor clicks the “Application” button on the left side tab. 2. Application Dashboard will be shown on UI. 3. Actor click on one of the applications shown on dashboard. 4. A Review Application dialog contain detailed information about that application will be shown. |
| Alternative flow |  |
| Post-condition | The system is ready for the next operation. |
| Extend point | None |

## Use-case “Find Loan Application”

|  |  |
| --- | --- |
| Use case’s name | Find Loan Application |
| Actor | Support Department, Business Department, Appraisal Department, Board of Director |
| Pre-condition | The actor must be authenticated |

|  |  |
| --- | --- |
| Basic flow | 1. Actor clicks the Application button on the left side tab. 2. Application Dashboard will be shown 3. Actor enter the information need to find (application number, customer phone,…) 4. List of contract result will be shown on UI. |
| Alternative flow | Found nothing => application will show empty state. |
| Post-condition | The system is ready for the next operation. |
| Extend point | None |

## Use-case “Reject Loan Application”

|  |  |
| --- | --- |
| Use case’s name | Reject Loan Application |
| Actor | Board of Director |
| Pre-condition | * The actor must be authenticated. * The use case “Review Loan Application” has just happened. * The Application has not been approved. |
| Basic flow | 1. Actor click “Reject” button. 2. The application will be rejected |
| Alternative flow |  |
| Post-condition | The system is ready for the next operation. |
| Extend point | None |

## Use-case “Approve Loan Application”

|  |  |
| --- | --- |
| Use case’s name | Approve Loan Application |
| Actor | Board of Director |
| Pre-condition | * The actor must be authenticated. * The use case “Review Loan Application” has just happened. * The Application has not been approved. |
| Basic flow | 1. Actor click “Approve” button. 2. The application will be Approved. |
| Alternative flow |  |
| Post-condition | The system is ready for the next operation. |
| Extend point | None |

## Use-case “Create customer”

|  |  |
| --- | --- |
| Use case’s name | Create Customer |
| Actor | Support Department, Business Department, Appraisal Department, Board of Director |
| Pre-condition | The actor must be authenticated |
| Basic flow | 1. Actor clicks the Application button on the left side tab. 2. Application Dashboard will be shown. 3. Actor click “Create” button. 4. A create dialog will be shown. |

|  |  |
| --- | --- |
|  | 1. Actor enter the required information. 2. Actor click save. 3. The customer will be created |
| Alternative flow | 6.1) The information is incorrect (missing some field, invalid format)  => System will show error dialog. |
| Post- condition | The system is ready for the next operation. |
| Extend point | None |

* 1. Use-case “Edit Customer Information”

|  |  |
| --- | --- |
| Use case’s name | Edit Customer Information |
| Actor | Support Department, Business Department, Appraisal Department, Board of Director |
| Pre-condition | * The actor must be authenticated. * The use case “Review Customer” has just happened. |
| Basic flow | 1. Actor click “Edit” button. 2. A edit customer screen will be shown. 3. Actor changes customer info. 4. Actor click “Save” button. 5. Customer will be updated. |
| Alternative flow | 4.1) The information is incorrect (missing some field, invalid format)  => System will show error dialog. |

|  |  |
| --- | --- |
| Post-condition | The system is ready for the next operation. |
| Extend point | None |

## Use-case “Review customer”

|  |  |
| --- | --- |
| Use case’s name | Review Customer |
| Actor | Support Department, Business Department, Appraisal Department, Board of Director |
| Pre-condition | The actor must be authenticated |
| Basic flow | 1. Actor clicks the “Customer” button on the left side tab. 2. List of Customer will be shown on UI. 3. Actor click on one of the Customer form that list 4. Detail information about that Customer will be shown |
| Alternative flow |  |
| Post-condition | The system is ready for the next operation. |
| Extend point | None |

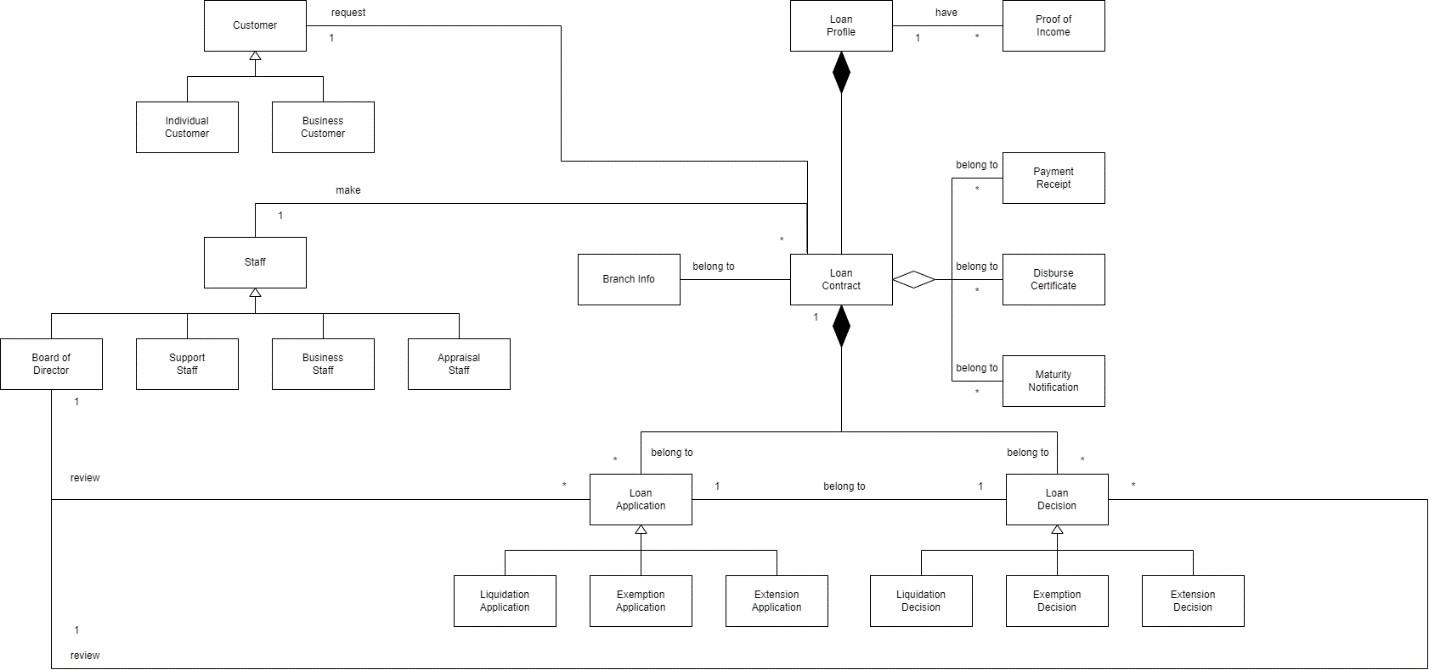
## Use-case “Find customer”

|  |  |
| --- | --- |
| Use case’s name | Find Customer |
| Actor | Support Department, Business Department, Appraisal Department, Board of Director |

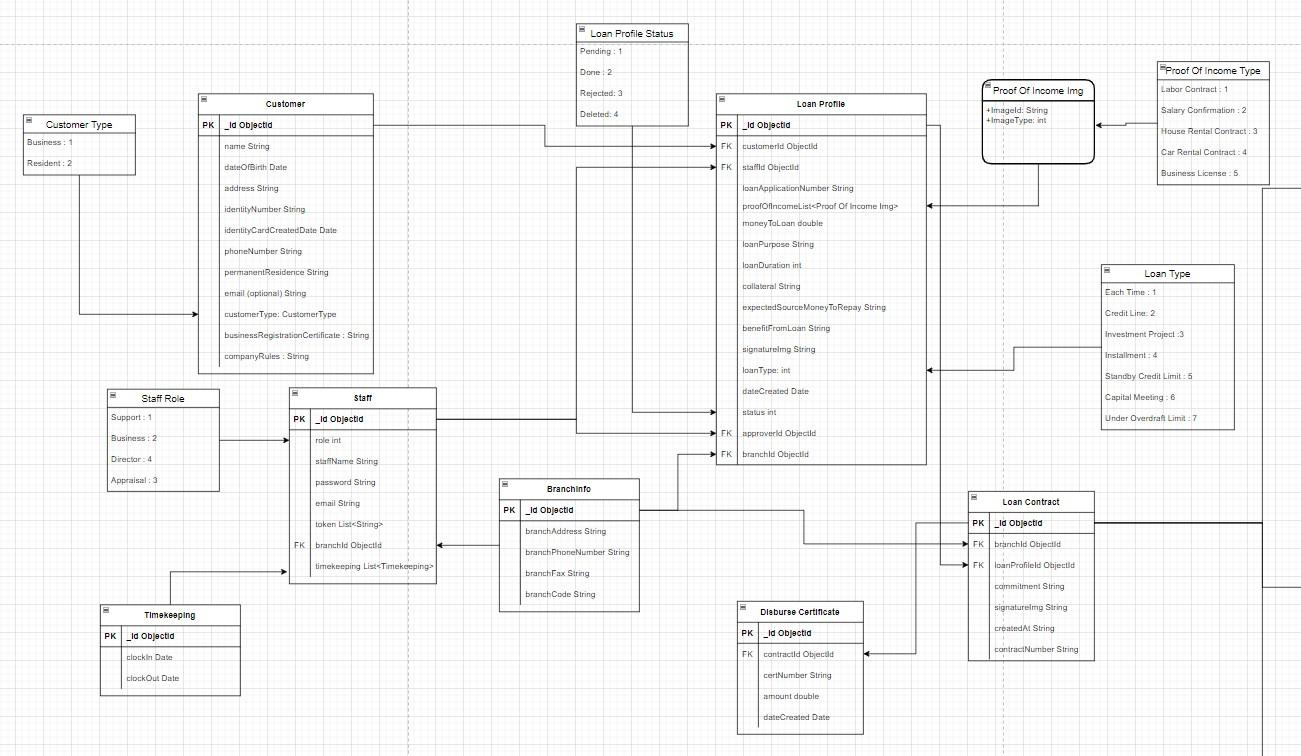
|  |  |
| --- | --- |
| Pre-condition | The actor must be authenticated |
| Basic flow | 1. Actor clicks the Customer button on the left side tab. 2. Customer Dashboard will be shown 3. Actor enter the information need to find (customer name, customer phone,…) 4. List of Customer result will be shown on UI. |
| Alternative flow | Found nothing => application will show empty state. |
| Post-condition | The system is ready for the next operation. |
| Extend point | None |

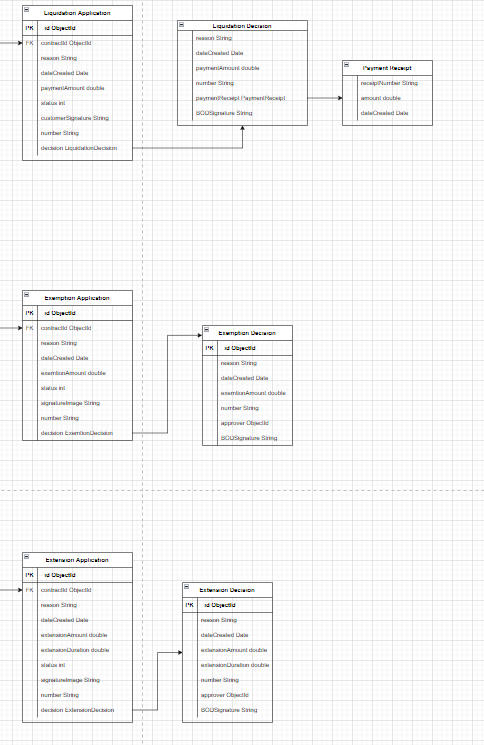
# Class diagram

## Class diagram (conceptual level)



## Class diagram (analysis level)





See full-size [here](https://drive.google.com/file/d/1ky7PFoDwExv8zNasMrBn14UfwrM-OMJv/view?usp=sharing)

## Class diagram explanation

## Loan Profile Table:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No | Attribute Name | Type | Constraints | Meaning/ Note |
| 1 | \_id | ObjectId | PK | id of loan profile, generated every time a new loan profile is created on mongoose. |
| 2 | customerId | ObjectId | FK | id of customer who owns this profile. |
| 3 | staffId | ObjectId | FK | id of staff who creates this profile. |
| 4 | loanApplicationNumber | String |  | identify number of this loan profile. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 5 | proofOfIncomeList | List<Proof of Income Img> |  | this loan profile’s proofs of income |
| 6 | moneyToLoan | Double |  | Loan amount of the loan profile |
| 7 | loanPurpose | String |  | Purpose of the loan profile |
| 8 | loanDuration | Int |  | Loan duration of the loan profile |
| 9 | collateral | String |  | Collateral of the loan profile |
| 10 | expectedSourceMoneyToRepay | String |  | Expected source of money use for repaying of the loan profile |
| 11 | benefitFromLoan | String |  | Bank’s benefit from this loan profile |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 12 | SignatureImg | String |  | Customer’s signature on loan profile |
| 13 | loanType | Int |  | Loan type of application |
| 14 | dateCreated | Date |  | Loan application creation date |
| 15 | Status | Int |  | Status of loan application |
| 16 | approverId | ObjectId | FK | id of the manager approves the profile |
| 17 | branchId | ObjectId | FK | Id of branch info where the loan application is created |

## Staff Table:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No | Attribute Name | Type | Constraints | Meaning/ Note |
| 1 | \_id | ObjectId | PK | id of each staff in the bank |
| 2 | role | Int |  | The role of the employee in the bank |
| 3 | staffName | String |  | Name of staff to display in the system |
| 4 | password | String |  | hashed-password of each employee used to log into the system, |
| 5 | email | String |  | Email of each employee used to log into the system |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 6 | token | List<String> |  | list of access tokens created when staff logged into the system |
| 7 | branchId | ObjectId | FK | Information about the branch where the employee works |
| 8 | timekeeping | List<Timekeeping> |  | employee timekeepings |

## Branch Info Table:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No | Attribute Name | Type | Constraints | Meaning/ Note |
| 1 | \_id | ObjectId | PK | id of each bank branch |
| 2 | branchAddress | String |  | address of each branch |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 3 | branchPhoneNumber | String |  | Phone number of each branch |
| 4 | branchFax | String |  | Fax of each branch |
| 5 | branchCode | String |  | Code of each branch |

## TimeKeeping Table:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No | Attribute Name | Type | Constraints | Meaning/ Note |
| 1 | \_id | ObjectId | PK | id of each timekeeping |
| 2 | clockIn | Date |  | time that staff starts working |
| 4 | clockOut | Date |  | time that staff ends working |

## Customer Table:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No | Attribute Name | Type | Constraints | Meaning/ Note |
| 1 | \_id | ObjectId | PK | id of each customer |
| 2 | name | String |  | Name of customer |
| 3 | dateOfBirth | Date |  | Date of birth of customer |
| 4 | address | String |  | Customer's address |
| 5 | identityNumber | String |  | Customer's identification number (ID) |
| 6 | identityCardCreatedDate | Date |  | customer’s identification card created date |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 7 | phoneNumber | String |  | Customer’s phone number |
| 8 | permanentResidence | String |  | customer's residence address |
| 9 | email | String | Optional | Customer’s email |
| 10 | customerType | Int |  | Customer type |
| 11 | businessRegistrationCertificate | String |  | Business’s certificate if customer is business customer |
| 12 | compayRules | String |  | Company rules if customer is business customer |

## Proof Of Income Img Table:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No | Attribute Name | Type | Constraints | Meaning/ Note |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | imageId | String |  | id of each image in loan application |
| 2 | imageType | Int |  | Proof of income type |

## Loan Contract Table:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No | Attribute Name | Type | Constraints | Meaning/ Note |
| 1 | \_id | ObjectId | PK | Id of each loan contract |
| 2 | branchId | ObjectId | FK | Id of branch info where the loan application is created |
| 3 | loanProfileId | ObjectId | FK | Id of loan profile related to this contract |
| 4 | commitment | String |  | Commitment of the loan contract |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 5 | signatureImg | String |  | Customer’s signature on loan contract |
| 6 | createdAt | String |  | Create at of loan contract |
| 7 | contractNumber | String |  | Contract number of loan contract |

## Disburse Certificate Table:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No | Attribute Name | Type | Constraints | Meaning/ Note |
| 1 | \_id | ObjectId | PK | Id of each disburse certificate |
| 2 | contractId | ObjectId | FK | Id of loan contract |
| 3 | certNumber | String |  | Number of certificates |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 4 | amount | Double |  | The amount specified in the certificate |
| 5 | dateCreated | Date |  | Date created disburse certificate |

## Liquidation Application Table:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No | Attribute Name | Type | Constraints | Meaning/ Note |
| 1 | \_id | ObjectId | PK | Id of Liquidation Application |
| 2 | contractId | ObjectId | FK | Id of loan contract |
| 3 | reason | String |  | Reason of liquidation application |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 4 | dateCreated | Date |  | Date Create liquidation application |
| 5 | paymentAmount | Double |  | the amount paid in the payment application |
| 6 | status | Int |  | Status of application |
| 7 | customerSignature | String |  | Customer’s signature in application |
| 8 | number | String |  | Number of liquidation application |
| 9 | decision | LiquidationDecision |  | Decision of the liquidation application |

## Liquidation Decision Table:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No | Attribute Name | Type | Constraints | Meaning/ Note |
| 1 | reason | String |  | Reason of liquidation decision |
| 2 | dateCreated | Date |  | Date created of liquidation decision |
| 3 | paymentAmount | Double |  | Payment amount of liquidation decision of the bank |
| 4 | number | String |  | Number of liquidation decision |
| 5 | paymentReceipt | PaymentReceipt |  | Payment receipt of liquidation decision |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 6 | approver | ObjectId |  | id of director creates the decision |
| 7 | BODSignature | String |  | signature url of director creates the decision |

## Payment Receipt Table:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No | Attribute Name | Type | Constraints | Meaning/ Note |
| 1 | receiptNumber | String |  | Number of receipt |
| 2 | amount | double |  | Amount of receipt |
| 3 | dateCreated | Date |  | Date created of payment receipt |

## Exemption Application Table:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No | Attribute Name | Type | Constraints | Meaning/ Note | |
| 1 | \_id | ObjectId | PK | Id of each application | exemption |
| 2 | contractId | ObjectId | FK | Id of loan contract | |
| 3 | reason | String |  | Reason of application | exemption |
| 4 | dateCreated | Date |  | Date created of exemption application | |
| 5 | exemptionAmount | Double |  | Amount of application | exemption |
| 6 | status | int |  | Status of application | exemption |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 7 | signatureImage | String |  | Customer’s signature on application |
| 8 | number | String |  | Number of exemption application |
| 9 | decision | Exemption Decision |  | Exemption Decision of the bank |

## Exemption Decision Table:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No | Attribute Name | Type | Constraints | Meaning/ Note |
| 1 | \_id | ObjectId | PK | Id of each exemption decision |
| 2 | reason | String |  | Reason of each exemption decision |
| 3 | dateCreated | Date |  | Date created of exemption decision |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 4 | exemptionAmount | Double |  | Amount of exemption decision |
| 5 | number | String |  | Number of exemption decision |
| 6 | approver | ObjectId |  | id of director creates the decision |
| 7 | BODSignature | String |  | signature url of director creates the decision |

## Extension Application Table:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No | Attribute Name | Type | Constraints | Meaning/ Note |
| 1 | \_id | ObjectId | PK | Id of each extension application |
| 2 | contractId | ObjectId | FK | Id of loan contract owns this application |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 3 | reason | String |  | Reason of extension application |
| 4 | dateCreated | Date |  | Date created of extension application |
| 5 | extensionAmount | Double |  | Amount of extension application |
| 6 | extensionDuration | Double |  | Duration of extension application |
| 7 | status | int |  | Status of extension application |
| 8 | signatureImage | String |  | Customer’s signature on application |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 9 | number | String |  | Number of extension application |
| 10 | decision | Extension Decision |  | extension Decision of the bank |

## Extension Decision Table:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No | Attribute Name | Type | Constraints | Meaning/ Note |
| 1 | \_id | ObjectId | PK | Id of each extension decision |
| 2 | reason | String |  | Reason of each extension decision |
| 3 | dateCreated | Date |  | Date created of extension decision |
| 4 | extensionAmount | Double |  | Amount of extension decision |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 5 | extensionDuration | Double |  | Duration of extension decision |
| 6 | number | String |  | Number of extension decision |
| 7 | approver | ObjectId |  | id of director creates the decision |
| 8 | BODSignature | String |  | signature url of director creates the decision |

## Enum Table:

|  |  |
| --- | --- |
| No | Loan Profile Status |
| 1 | Pending |
| 2 | Done |
| 3 | Rejected |
| 4 | Deleted |

|  |  |
| --- | --- |
| No | Proof of Income Type |
| 1 | Labor Contract |
| 2 | Salary Confirmation |
| 3 | House Rental Contract |
| 4 | Car Rental Contract |
| 5 | Business License |

|  |  |
| --- | --- |
| No | Loan Type |
| 1 | Each Time |
| 2 | Credit Line |
| 3 | Investment Project |
| 4 | Installment |

|  |  |
| --- | --- |
| 5 | Standby Credit Limit |
| 6 | Capital Meeting |
| 7 | Under Overdraft Limit |

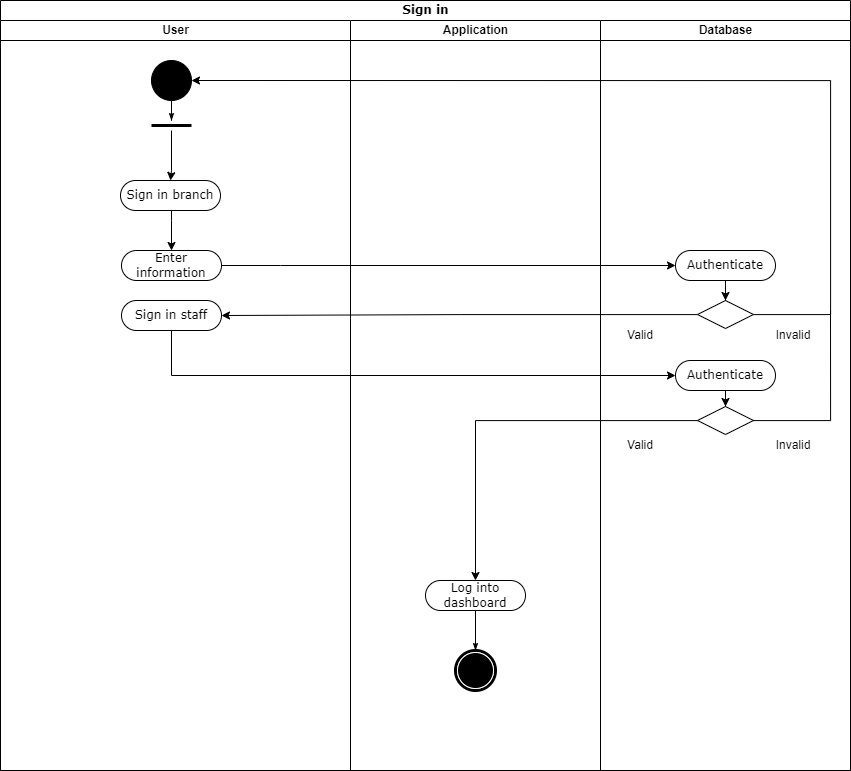
|  |  |
| --- | --- |
| No | Customer Type |
| 1 | Business |
| 2 | Resident |

|  |  |
| --- | --- |
| No | Staff Role |
| 1 | Support |
| 2 | Business |
| 3 | Director |

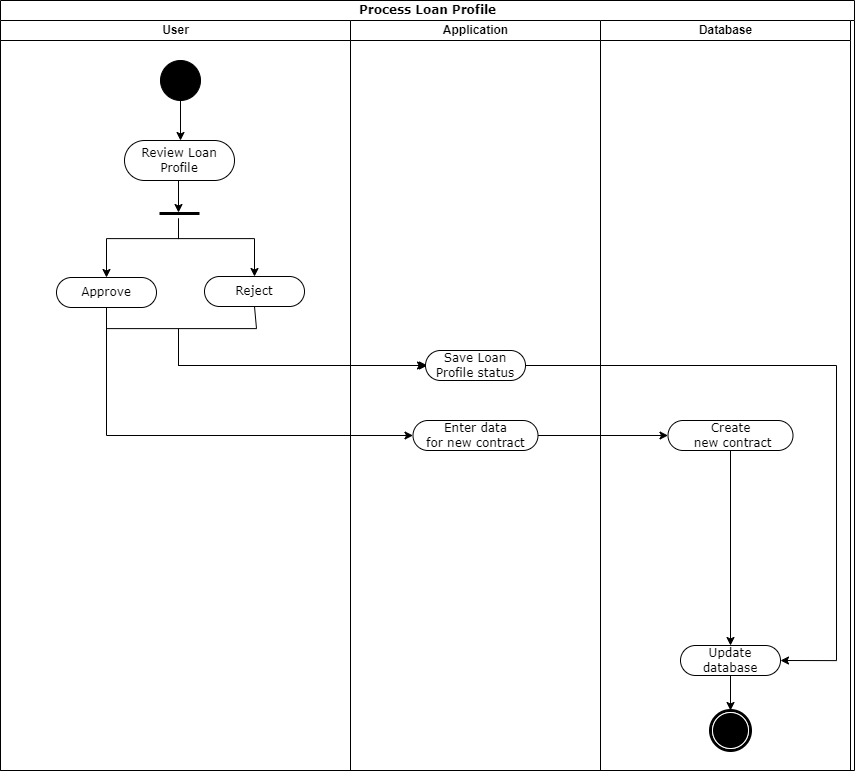
|  |  |
| --- | --- |
| 4 | Appraisal |

# Activity diagram

* 1. Sign in diagram

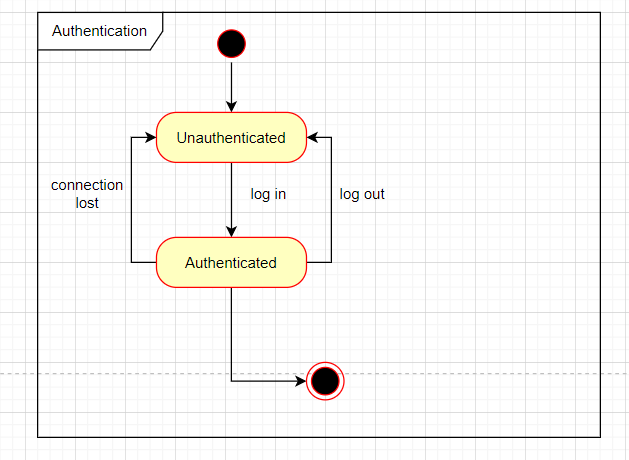


## Process loan profile diagram

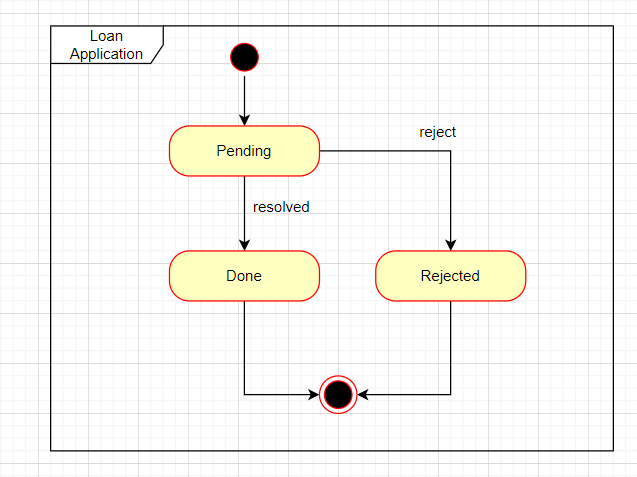


# State diagram

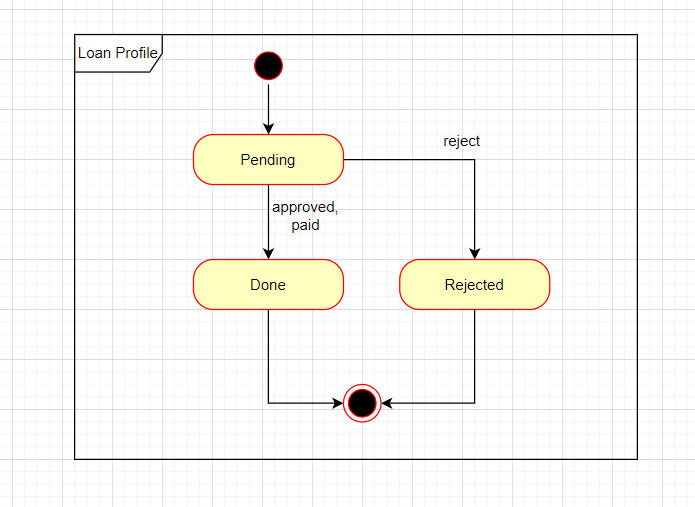
## Authentication status diagram



## Loan application status diagram

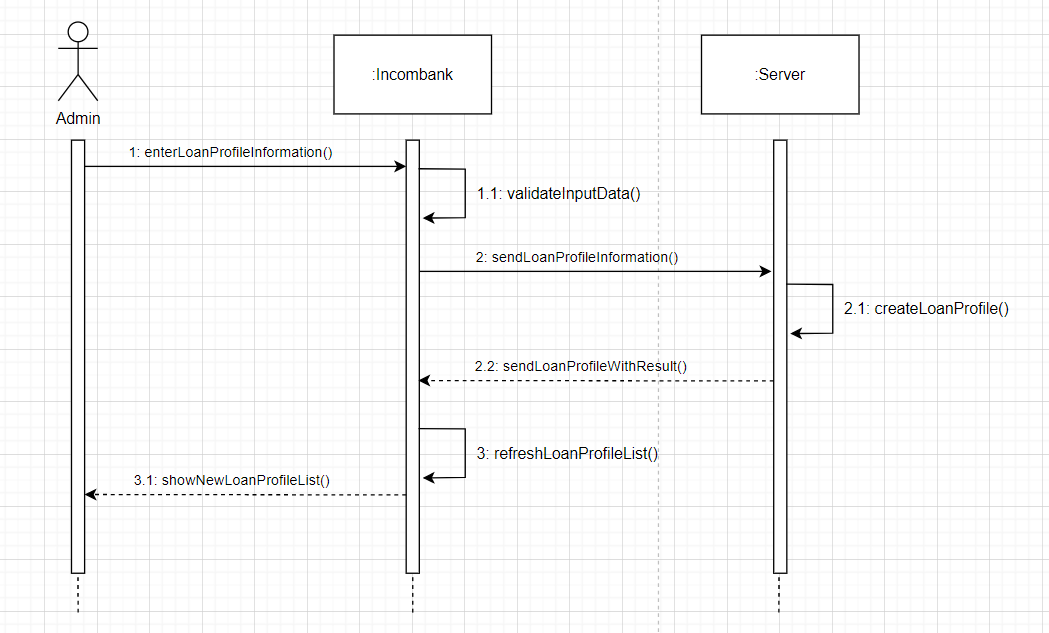


## Loan profile status diagram

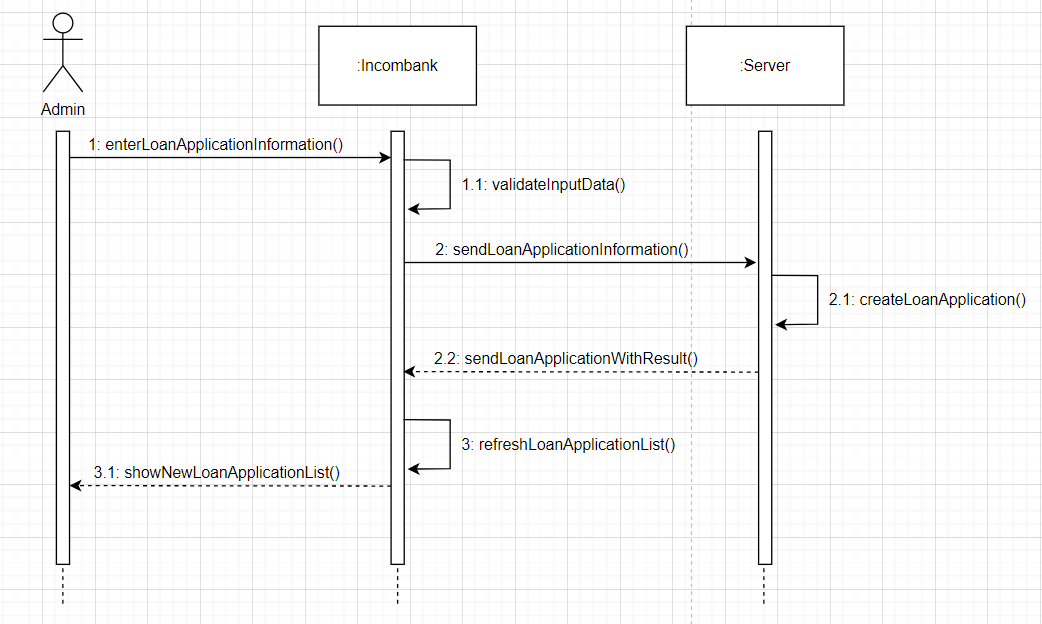


# Sequence diagram

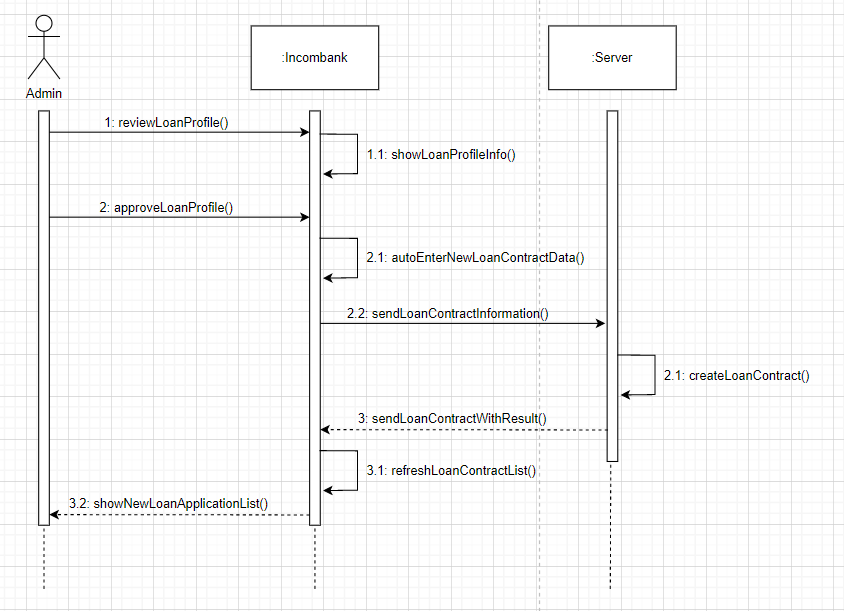
## Create loan profile diagram



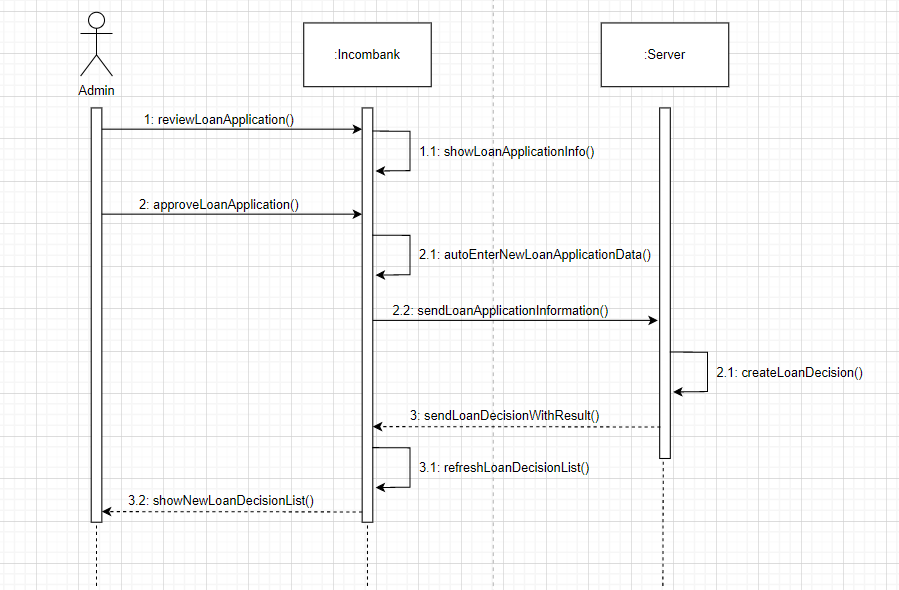
## Create loan application diagram



## Create loan contract diagram

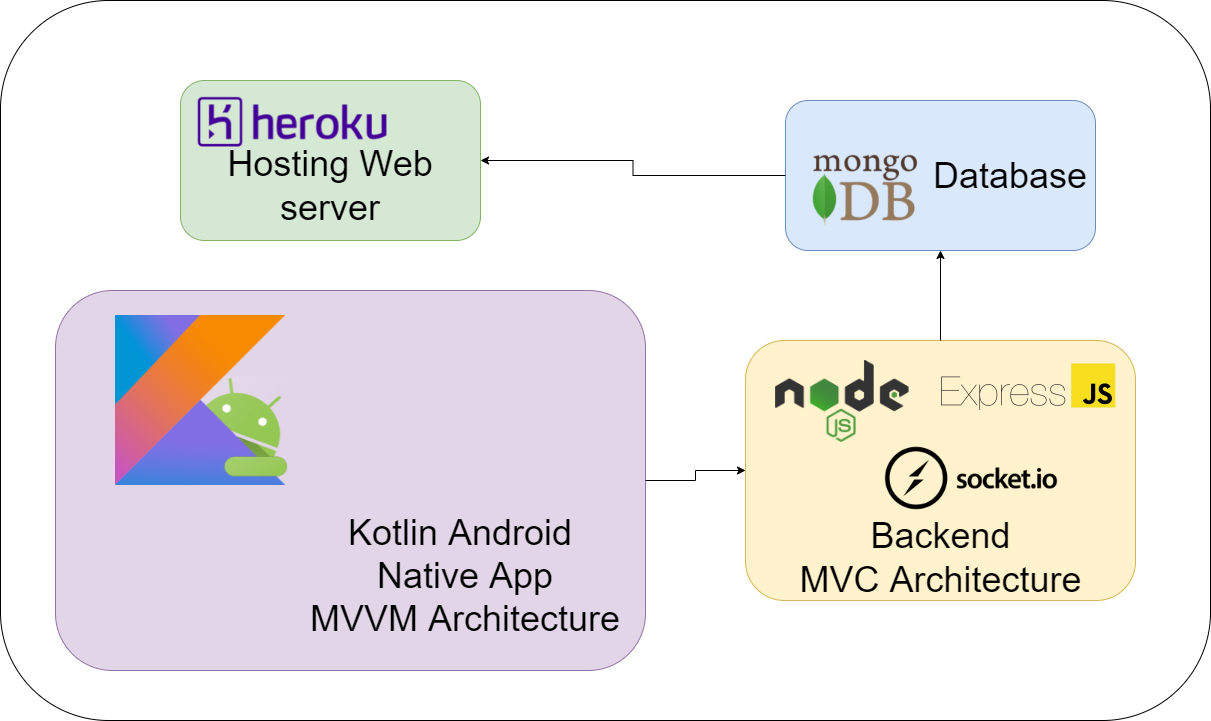


## Create loan decision diagram



# Architecture design

## Application Architecture Overview

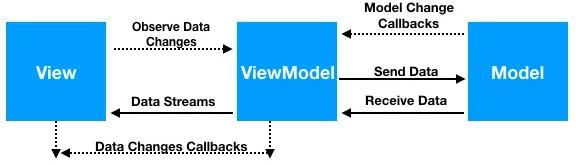


This application is implemented with the architecture shown above, it includes:

* *Front-end:* is the layer in which the users interact with an application. Front-end contents Shared UI code, Code Behind and Designers used to represent information to user.This layer is built based on Android framework,written in Kotlin and target tablet devices, with MVVM as its architecture.
* *Back-end:* is the layer responsible for providing RestAPI for the front-end layer, as well as handling the communication between front-end and database.This layer is built on top of NodeJS, ExpressJS and socket.io, written in Javascript with MVC architecture.
* *Database:* This application use mongoDB as its database, it’s a is a document database designed for ease of development and scaling
* *Hosting:* The back-end layer is hosted on Heroku, it’s a free hosting service for small project, so it’s quite slow for performance.

## Application Architecture in detail

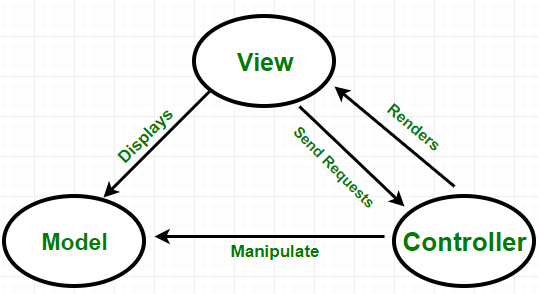
## Front-end layer



The front-end layer is built with MVVM (Model-View-View Model) architecture, which includes :

* + - *Model:* This holds the data of the application. It cannot directly talk to the View. Generally, it’s recommended to expose the data to the ViewModel through Observables.
    - *View:* It represents the UI of the application devoid of any Application Logic. It observes the ViewModel.
    - *ViewModel:* It acts as a link between the Model and the View. It’s responsible for transforming the data from the Model. It provides data streams to the View. It also uses hooks or callbacks to update the View. It’ll ask for the data from the Model.

## Back-end layer

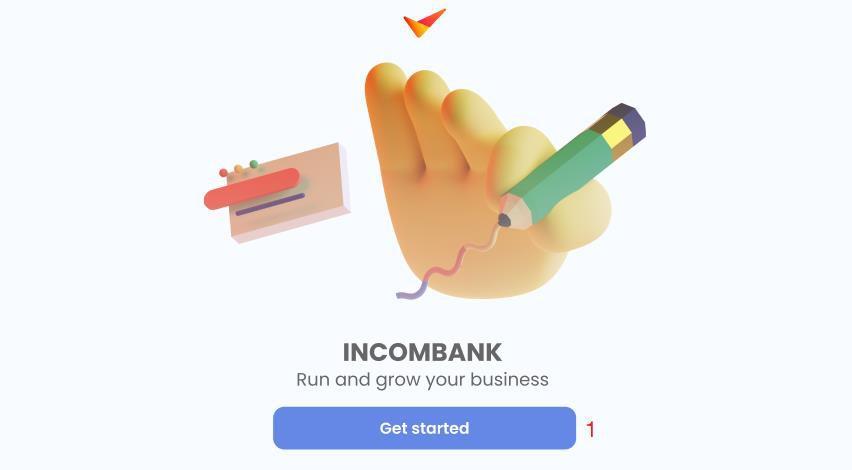


The back-end layer is built with MVC (Model-View-Controller) architecture, which includes :

* + - Model: Model represents the structure of data, the format and the constraints with which it is stored. It maintains the data of the application. Essentially, it is the database part of the application.
    - View: View is what is presented to the user. Views utilize the Model and present data in a form in which the user wants. A user can also be allowed to make changes to the data presented to the user. They consist of static and dynamic pages which are rendered or sent to the user when the user requests them.
    - Controller: Controller controls the requests of the user and then generates appropriate response which is fed to the viewer. Typically, the user interacts with the View, which in turn generates the appropriate request, this request will be handled by a controller. The controller renders the appropriate view with the model data as a response.

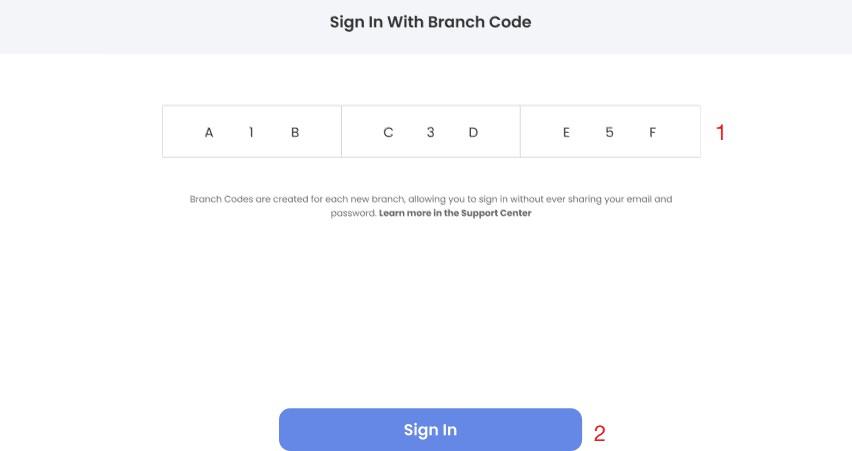
# Interface design

## Welcome screen



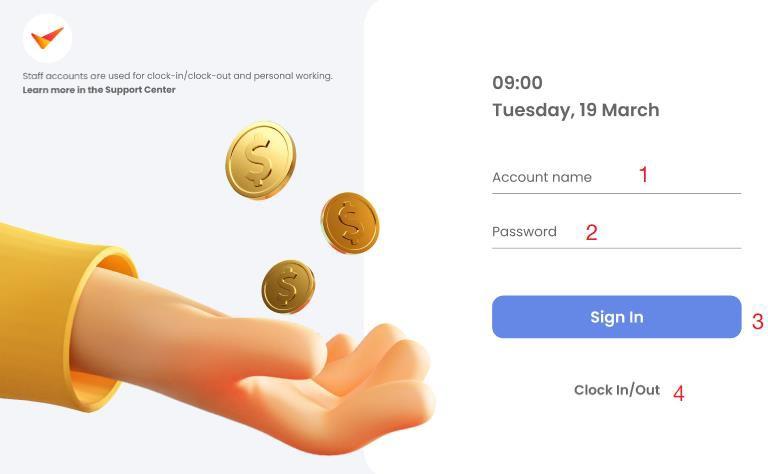
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No | Name | Type | Feature | Calling Condition | Note |
| 1 | getStarted\_Button | Button | Navigate to  sign in branch screen | Click the button |  |

## Sign in brach screen



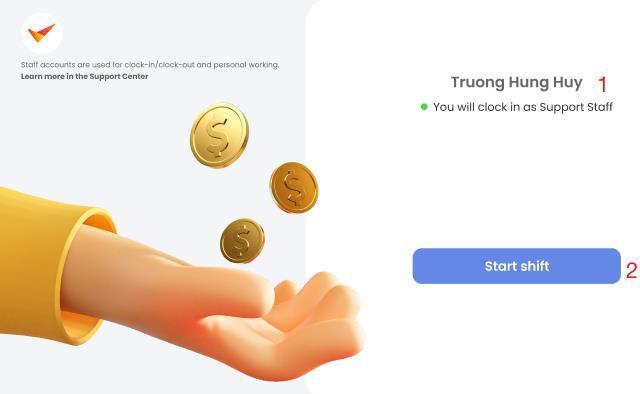
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No | Name | Type | Feature | Calling Condition | Note |
| 1 | branchCode\_EditText | EditText | Enter input as  branch code | Type in edit text |  |
| 2 | getStarted\_Button | Button | Navigate to sign in staff  screen | Click the button |  |

## Sign in staff screen



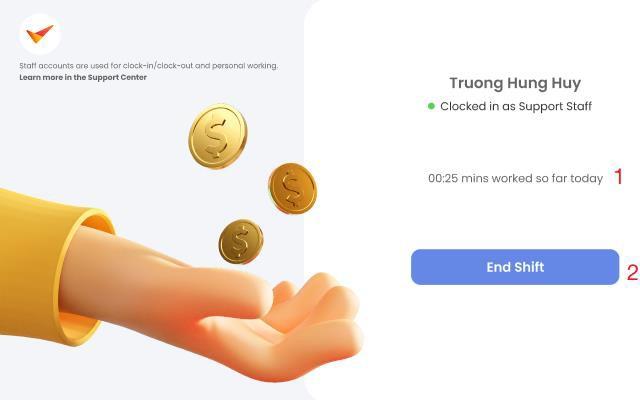
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No | Name | Type | Feature | Calling Condition | Note |
| 1 | account\_EditText | EditText | Enter input as  account | Type in edit text |  |
| 2 | password\_EditText | EditText | Enter input as password | Type in edit text |  |
| 3 | signIn\_Button | Button | Navigate to dashboard if not in “Clock  in/out” mode | Click the button |  |
| 4 | clockInOut\_Button | Button | Enable “Clock in/out” mode | Click the button |  |

## Clock in/out screen (start shift)



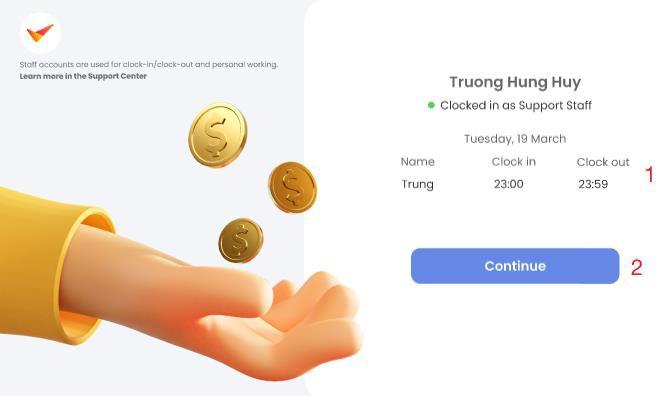
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No | Name | Type | Feature | Calling Condition | Note |
| 1 | staffInfo\_TextView | TextView | Show info |  |  |
| 2 | startShift\_ Button | Button | Navigate | Click the button |  |

## Clock in/out screen (end shift)



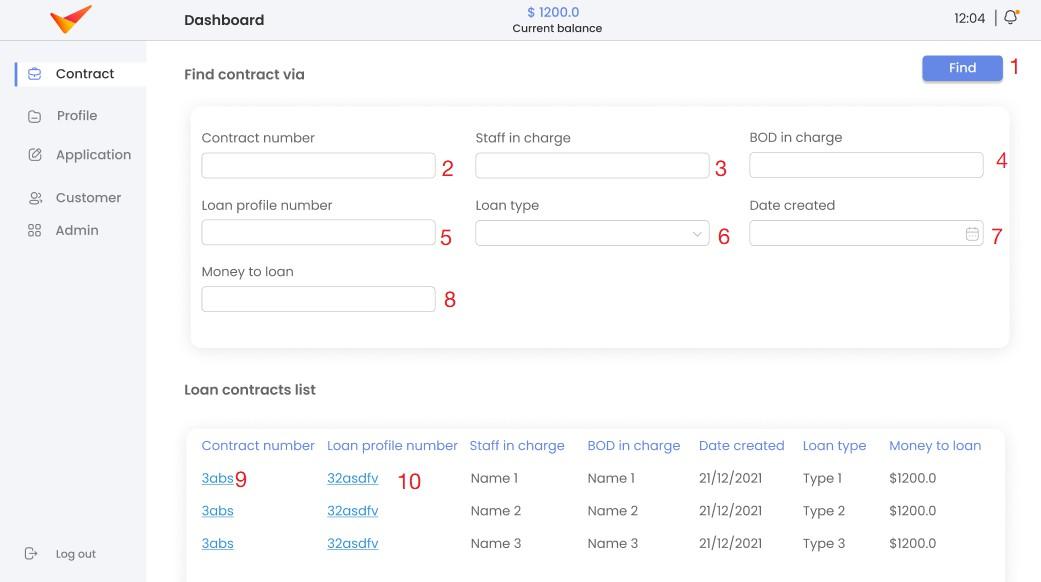
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No | Name | Type | Feature | Calling Condition | Note |
| 1 | staffTimekeeping\_TextView | TextView | Show info |  |  |
| 2 | endShift\_ Button | Button | Navigate | Click the button |  |

## Clock in/out screen (Summarize)



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No | Name | Type | Feature | Calling Condition | Note |
| 1 | staffTimekeepingResult\_TextView | TextView | Show  info |  |  |
| 2 | continue\_ Button | Button | Navigate to sign in  staff screen | Click the button |  |

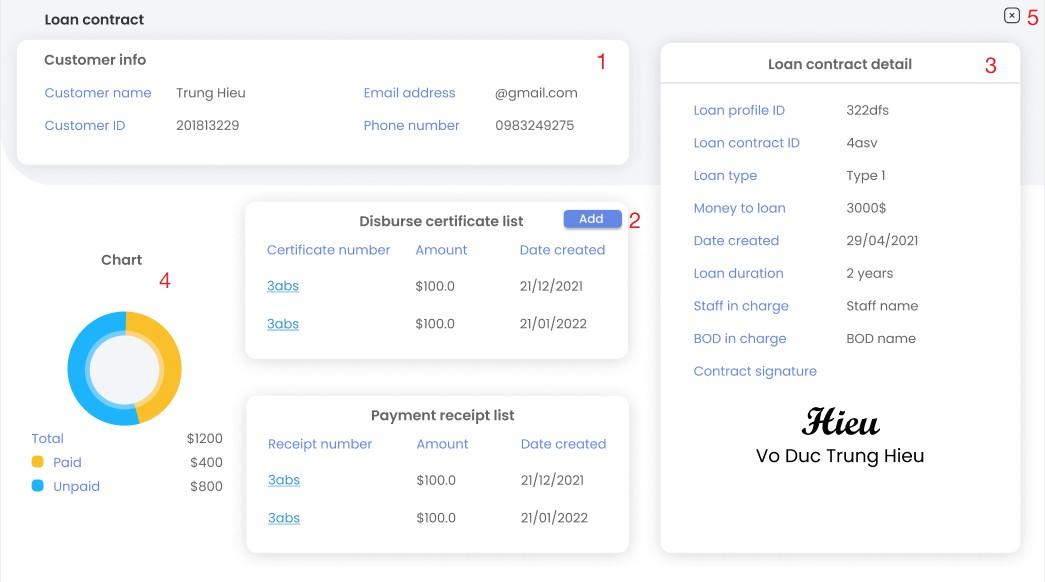
## Contract screen



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| .No | Name | Type | Feature | Calling Conditions | Note |
| 1 | Find\_Button | Button | Find contract | Click on button | Information needed to search |
| 2 | Contract  number\_Edit Text | Edit Text | Enter  contract number | Fill in edit text |  |
| 3 | Staff in charge\_Edit Text | Edit Text | Enter Staff in charge | Fill in edit text |  |
| 4 | BOD in charge\_Edit Text | Edit Text | Enter BOD in charge | Fill in edit text |  |
| 5 | Loan Profile number\_Edit  Text | Edit Text | Enter Loan profile  number | Fill in edit text |  |

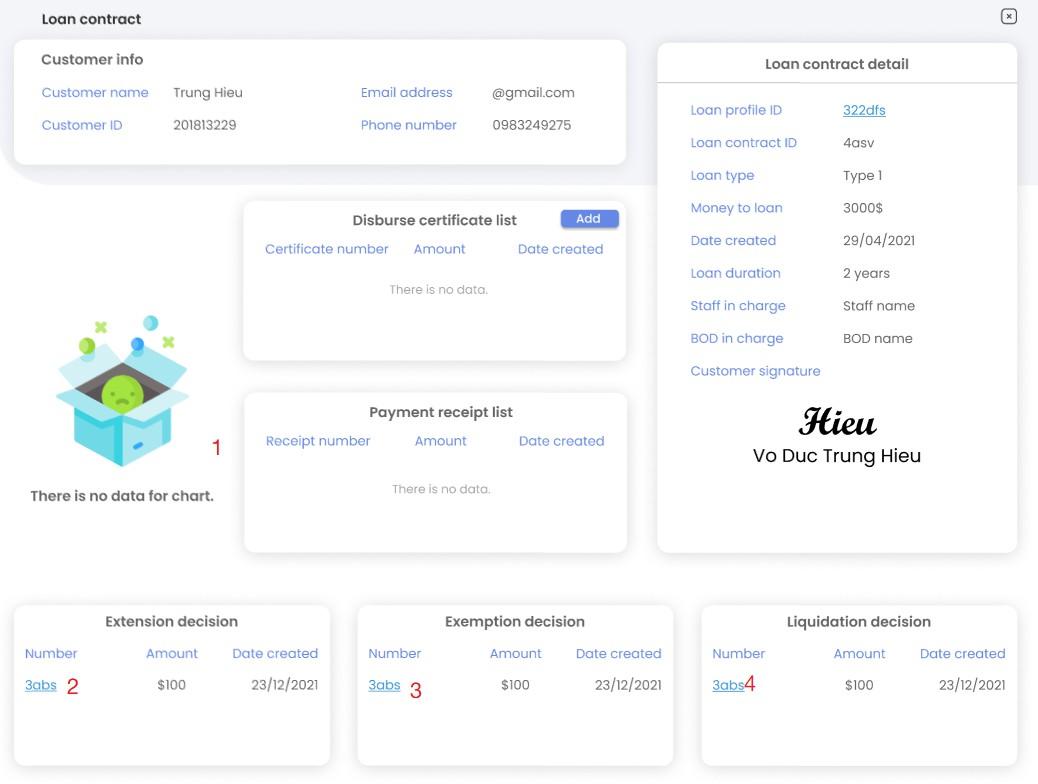
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 6 | Loan type\_ComboBox | Combo Box | Select loan type | Select child  elements of combo box |  |
| 7 | Date created\_Button | Button | Select Date created | Click on button calendar |  |
| 8 | Money to loan\_Edit text | Edit Text | Enter Money to loan | Fill in edit text |  |
| 9 | Contract item\_TextButton | Text Button | Navigate to detail screen | Click on button | Navigate to Loan  Contract Detail |
| 10 | Profile item\_TextButton | Text Button | Navigate to detail screen | Click on button | Navigate to Loan Profile  Detail |

* 1. Contract detail



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No | Name | Type | Feature | Calling Condition | Note |
| 1 | customer\_CardView | CardView | Show data |  |  |
| 2 | addDisburse\_Button | Button | Add disburse | Click the button |  |
| 3 | loanContract\_CardView | CardView | Show data |  |  |
| 4 | chart | Chart | Show data |  |  |
| 5 | Close\_Button | Button | Navigate to dashboard | Click the button |  |

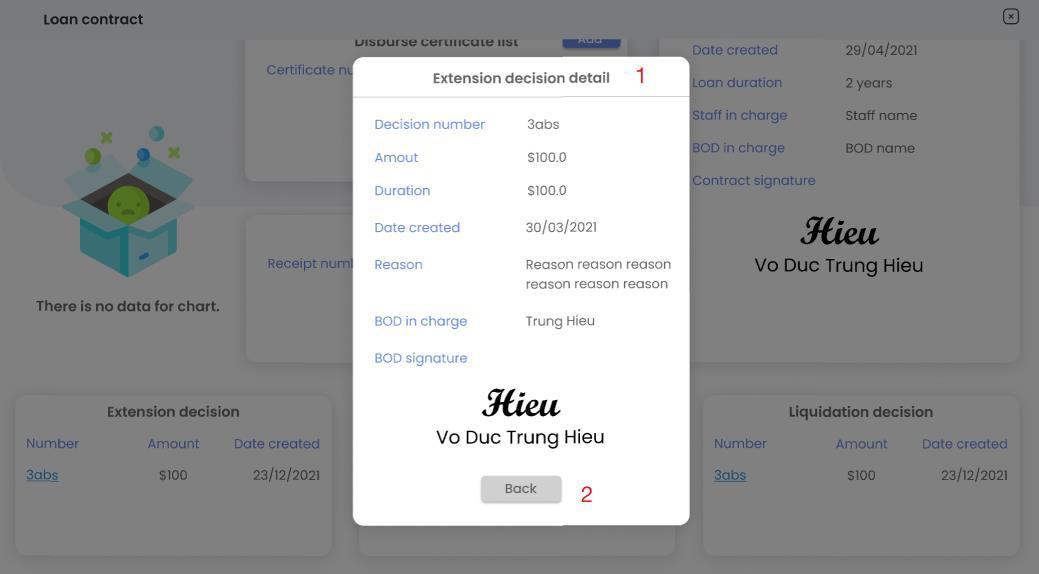
## Contract detail (below)



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No | Name | Type | Feature | Calling Condition | Note |
| 1 | emptyState\_ImageView | ImageView | Show empty  state | Only show when  there is no data |  |
| 2 | extension item\_TextButton | Text Button | Navigate to detail screen | Click on button |  |
| 3 | Exemption item\_TextButton | Text Button | Navigate to detail screen | Click on button |  |

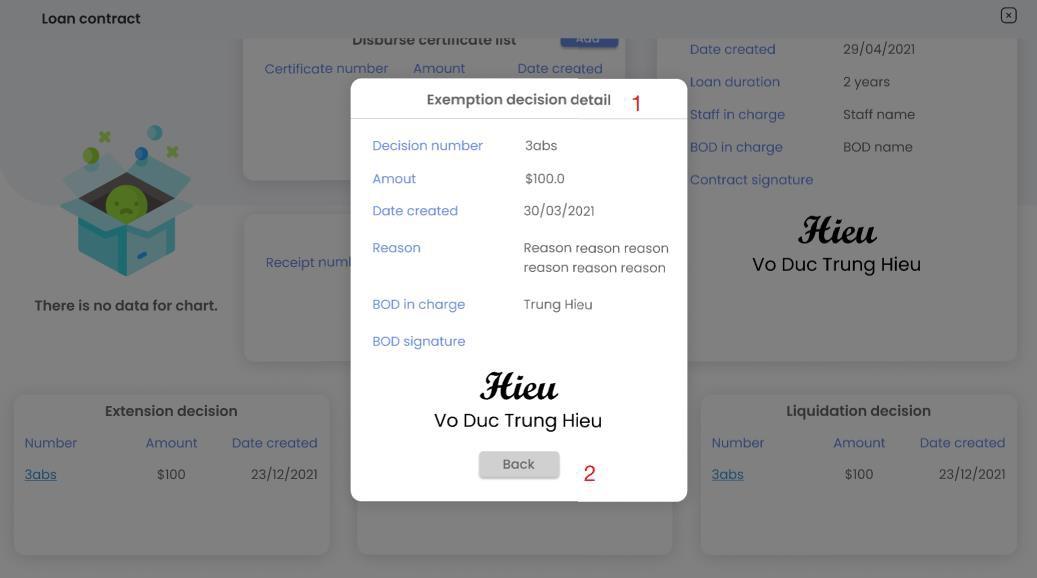
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 4 | Liquidation item\_TextButton | Text Button | Navigate to detail screen | Click on button |  |

## Extension decision detail screen



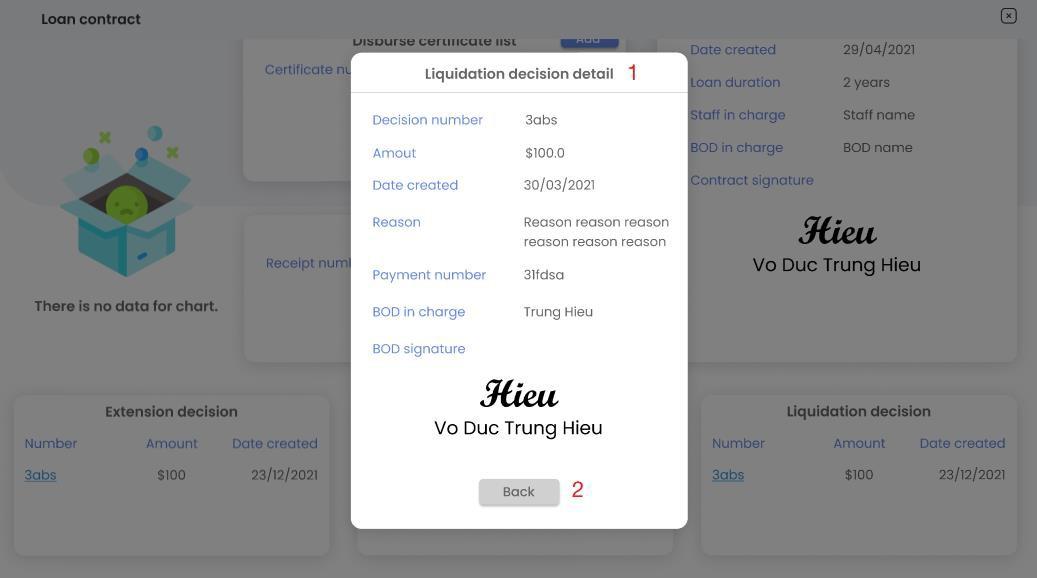
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No | Name | Type | Feature | Calling Condition | Note |
| 1 | detail\_CardView | CardView | Show data |  |  |
| 2 | Back\_Button | Button | Navigate one back | Click the button |  |

## Exemption decision detail screen



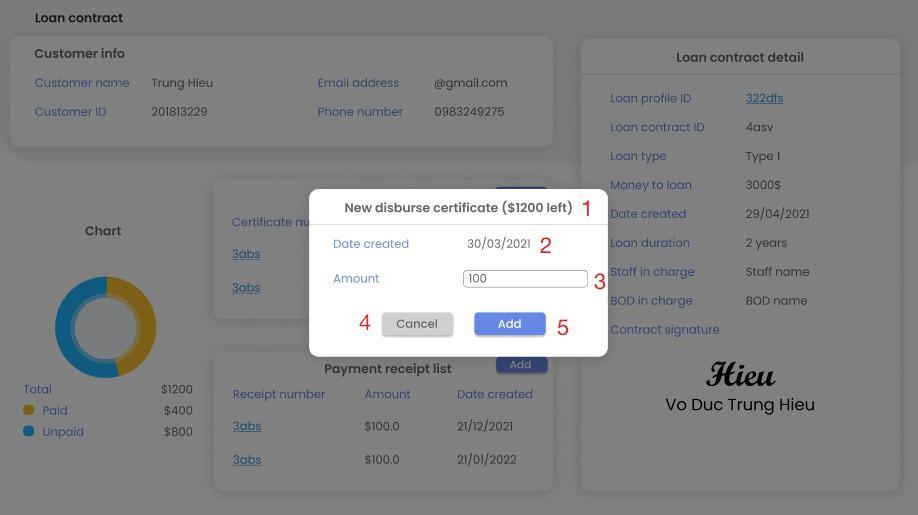
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No | Name | Type | Feature | Calling Condition | Note |
| 1 | detail\_CardView | CardView | Show data |  |  |
| 2 | Back\_Button | Button | Navigate one back | Click the button |  |

## Liquidation decision detail screen



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No | Name | Type | Feature | Calling Condition | Note |
| 1 | detail\_CardView | CardView | Show data |  |  |
| 2 | Back\_Button | Button | Navigate one back | Click the button |  |

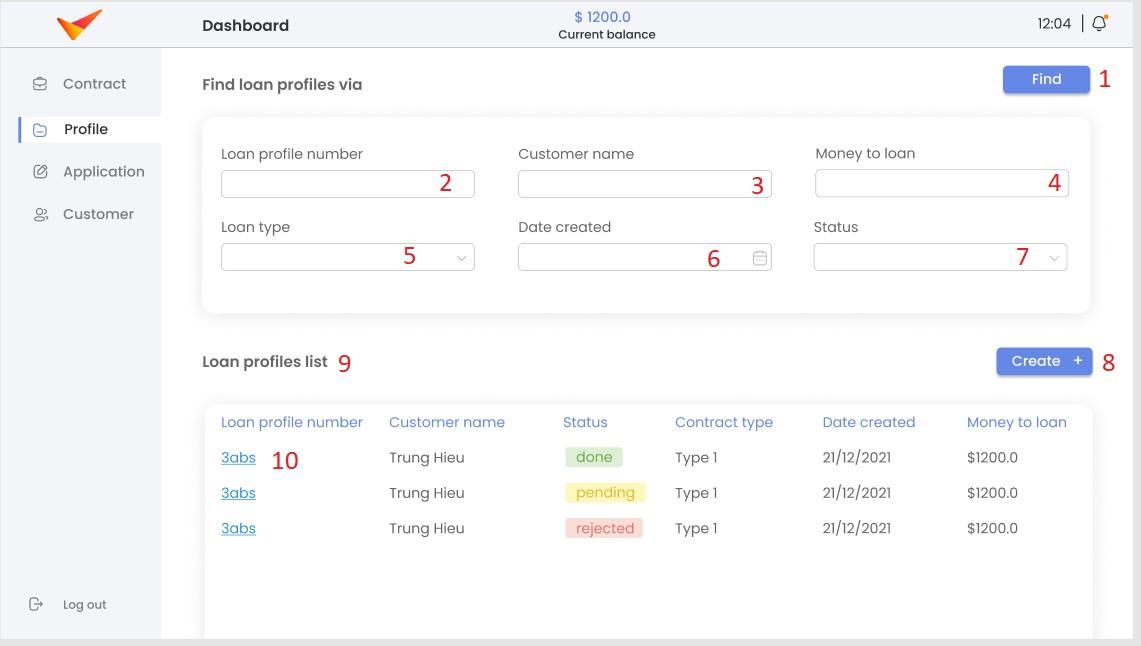
## Add disburse screen



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No | Name | Type | Feature | Calling Condition | Note |
| 1 | amountLeft\_TextView | TextView | Show data |  |  |
| 2 | dateCreated\_ TextView | TextView | Show data |  |  |
| 3 | amount\_EditText | EditText | Enter amount data | Enter in EditText |  |
| 4 | cancel\_Button | Button | Navigate one back | Click the button |  |
| 5 | add\_Button | Button | Add disburse  and go one back | Click the button |  |

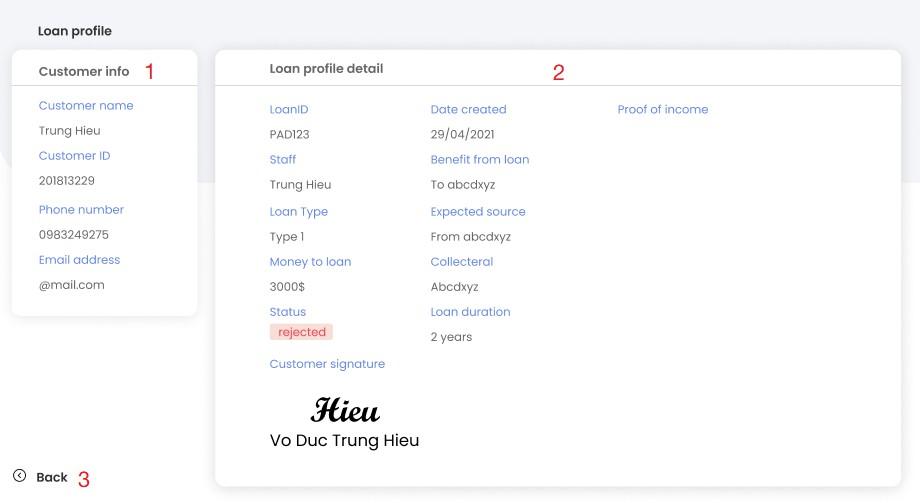
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No | Name | Type | Feature | Calling Condition | Note |
| 1 | account\_EditText | EditText | Enter input as  account | Type in edit text |  |
| 2 | password\_EditText | EditText | Enter input as password | Type in edit text |  |
| 3 | signIn\_Button | Button | Navigate to dashboard if not in “Clock  in/out” mode | Click the button |  |
| 4 | clockInOut\_Button | Button | Enable  “Clock in/out” mode | Click the button |  |

## Loan profile screen



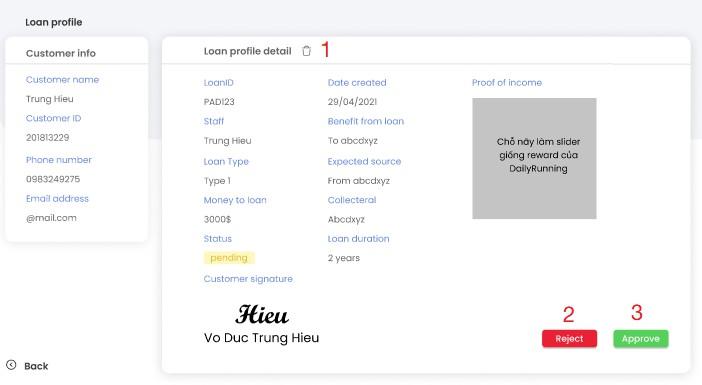
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No | Name | Type | Feature | Calling Conditions | Note |
| 1 | Find\_Button | Find Button | Find loan profile | Click on button |  |
| 2 | Loan profile number\_EditText | Edit Text | Enter Loan profile  number | Fill in Edit Text |  |
| 3 | Customer name\_EditText | EditText | Enter customer  name | Fill in Edit Text |  |
| 4 | Money to loan\_EditText | EditText | Enter Money to loan | Fill in Edit Text |  |
| 5 | Loan Type\_ComboBox | Combo Box | Select Loan Type | Select child elements of  combo box |  |
| 6 | Date created\_Button | Button | Select Date created | Click on button calendar |  |
| 7 | Loan  Status\_Combo Box | Combo Box | Select Loan Status | Select child  elements of combo box |  |
| 8 | Create new Loan Profile\_Button | Button | Create new Loan Profile | Click on button |  |
| 9 | Loan profile list\_RecyclerView | Recycler View | Show list of loan profile | Automatic | Change  when click button Find |
| 10 | Loan profile item\_TextButton | Text Button | Navigate to detail screen | Click on text button |  |

## Profile detail screen



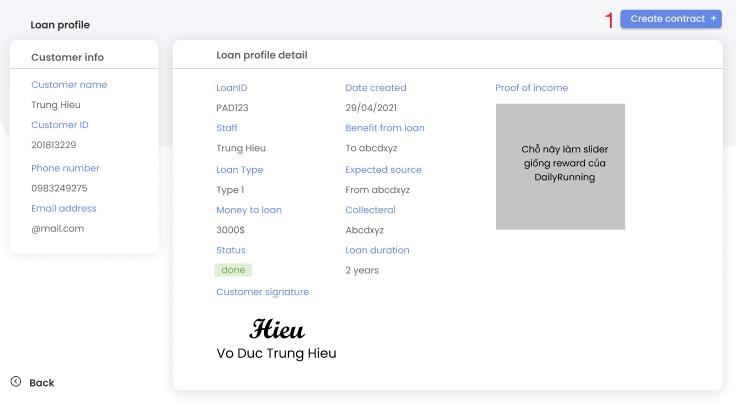
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No | Name | Type | Feature | Calling Condition | Note |
| 1 | customerInfo\_CardView | CardView | Show data |  |  |
| 2 | loanProfile\_ CardView | CardView | Show data |  |  |
| 3 | back\_Button | Button | Navigate to  profile screen | Click the button |  |

## Loan profile (Reject/Approve)



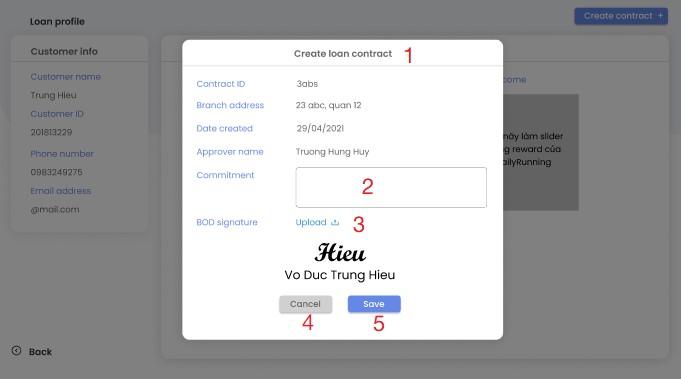
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No | Name | Type | Feature | Calling Condition | Note |
| 1 | delete\_Button | Button | Delete Loan profile | Click the button |  |
| 2 | reject\_Button | Button | Reject loan profile | Click the button |  |
| 3 | approve\_Button | Button | Approve loan profile | Click the button |  |

## Loan profile (Create contract)



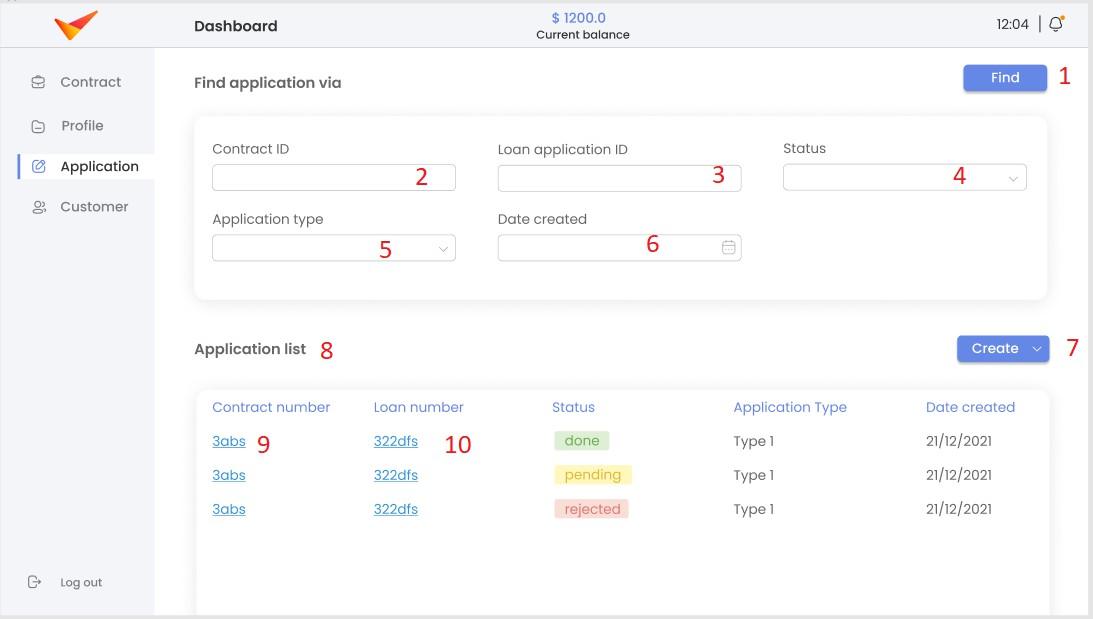
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No | Name | Type | Feature | Calling Condition | Note |
| 1 | createContract\_Button | Button | Create  contract | Clink the button |  |

## Create contract



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No | Name | Type | Feature | Calling Condition | Note |
| 1 | loanContract\_CardView | CardView | Show data |  |  |
| 2 | commitment\_EditText | EditText | Enter input  as commitment | Type in edit text |  |
| 3 | upload\_Button | Button | Upload signature  image | Click the button |  |
| 4 | cancel\_Button | Button | Navigate one back | Click the button |  |
| 5 | save\_Button | Button | Save loan contract | Click the button |  |

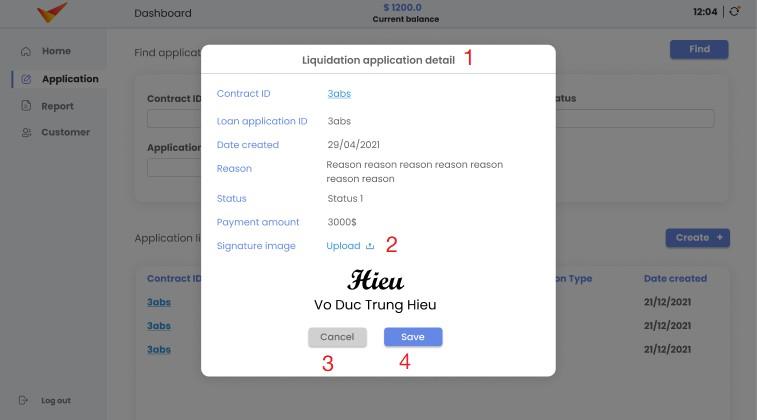
## Application screen



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No | Name | Type | Feature | Calling Conditions | Note |
| 1 | Find\_Button | Find Button | Find application | Click on button |  |
| 2 | Contract\_EditText | Edit Text | Enter Contract | Fill in Edit Text |  |
| 3 | Loan Application ID\_EditText | EditText | Enter Loan  Application ID | Fill in Edit Text |  |
| 4 | Loan Status\_Combo Box | Combo Box | Select Loan Status | Select child elements of  combo box |  |
| 5 | Loan Type\_ComboBox | Combo Box | Select Loan Type | Select child elements of  combo box |  |

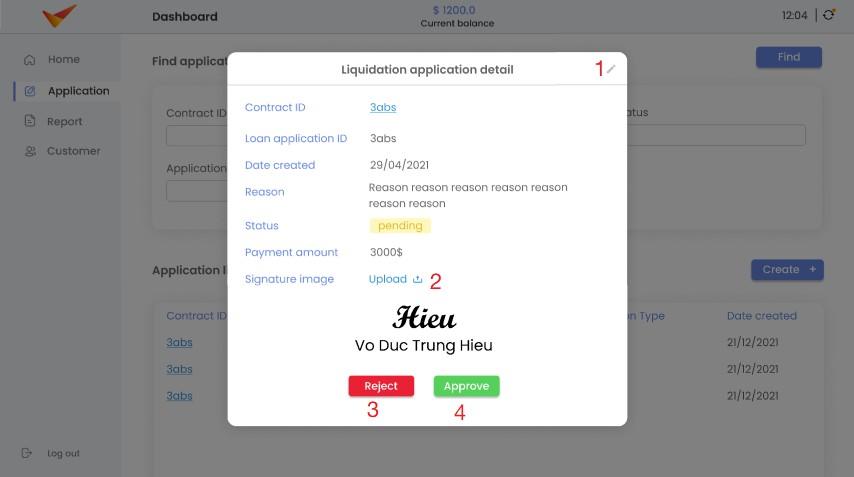
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 6 | Date created\_Button | Button | Select Date created | Click on button calendar |  |
| 7 | Create new Loan Application\_Button | Button | Create new Application | Click on button | Show Combo box to select  type create |
| 8 | Application List\_RecyclerView | Recycler View | Show list of application | Automatic | Change when click  button Find |
| 9 | Contract number item\_TextButton | Text Button | Show info base on contract  number | Click on text button | Navigate to Contract Detail  Screen |
| 10 | Loan number item\_TextButton | Text Button | Show info  base on Loan number | Click on text button | Navigate to  Loan Detail Screen |

## Application detail



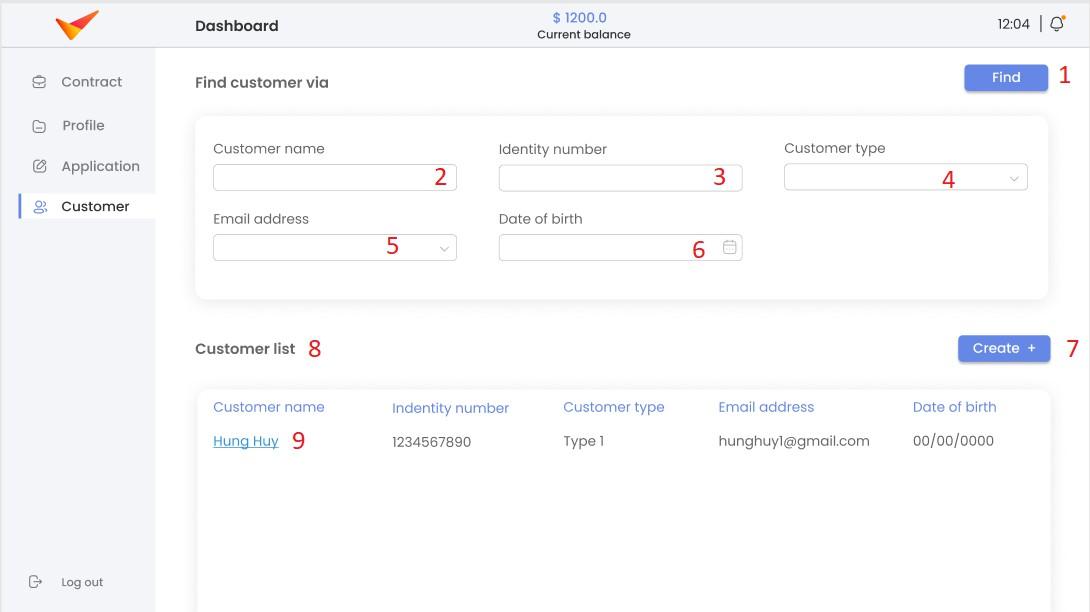
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No | Name | Type | Feature | Calling Condition | Note |
| 1 | loanApplication\_CardView | CardView | Show data |  |  |
| 2 | upload\_Button | Button | Upload  signature image | Click the button |  |
| 3 | cancel\_Button | Button | Navigate one back | Click the button |  |
| 4 | save\_Button | Button | Save loan contract | Click the button |  |

## Application detail screen (Reject/Approve)



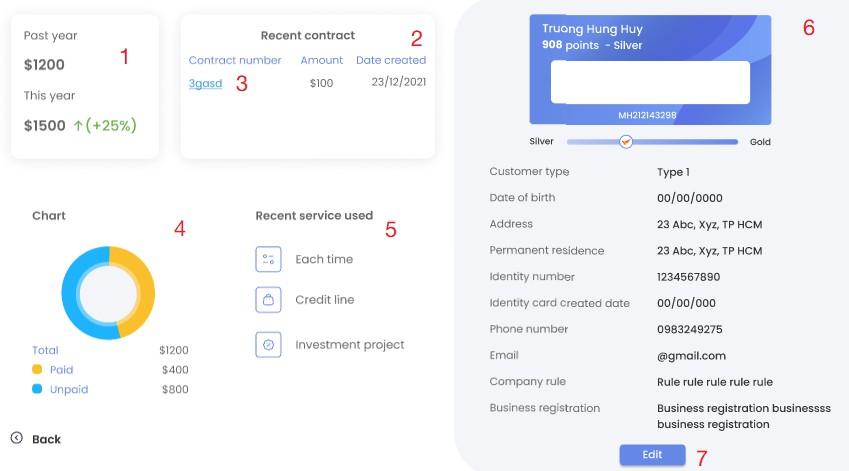
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No | Name | Type | Feature | Calling Condition | Note |
| 1 | edit\_Button | Button | Change to  edit mode | Type in edit text |  |
| 2 | upload\_Button | Button | Upload signature image | Click the button |  |
| 3 | reject\_Button | Button | Reject  application | Click the button |  |
| 4 | approve\_Button | Button | Approve application | Click the button |  |

## Customer screen



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| .No | Name | Type | Feature | Calling Conditions | Note |
| 1 | Find\_Button | Find Button | Find Customer | Click on button |  |
| 2 | Customer Name\_EditText | Edit Text | Enter customer  name | Fill in Text Edit |  |
| 3 | Identity number\_EditText | Edit Text | Enter Identity number | Fill in Text Edit |  |
| 4 | Customer Type\_ComboBox | Combo Box | Select  customer type | Select child  elements of combo box |  |
| 5 | Email Address\_EditText | Edit Text | Enter Emaill  Address of customer | Fill in Edit Text |  |
| 6 | Date of birth\_Button | Button | Select Date of birth | Click on button |  |
| 7 | Create new Customer\_Button | Button | Create new customer | Click on button | Navigate to Create new  Customer Screen |
| 8 | Customer List\_RecyclerView | Recycler View | Show list of customer in system | Automatic | Change when click on button  Find |
| 9 | Customer Item\_TextButton | TextButton | Show info of customer | Click on button | Navigate to Customer Detail  Screen |

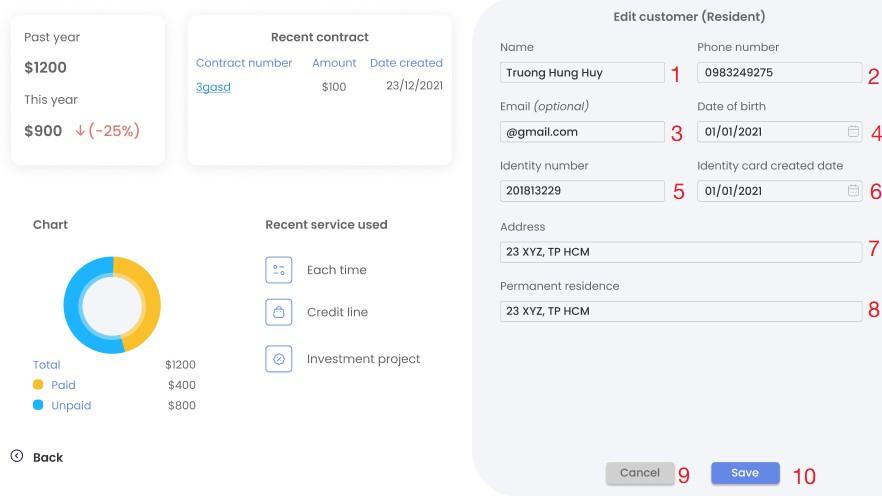
## Customer info screen



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No | Name | Type | Feature | Calling Condition | Note |
| 1 | amount\_CardView | CardView | Show  data |  |  |
| 2 | recentContract\_ CardView | CardView | Show data |  |  |
| 3 | contractNumber\_TextButton | TextButton | Navigate to contract  detail screen | Click the button |  |
| 4 | chart | Chart | Show data |  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 5 | recentServiceUsed\_RecyclerView | RecyclerView | Show data |  |  |
| 6 | customerInfo\_CardView | CardView | Show data |  |  |
| 7 | Edit\_Button | Button | Change to edit  mode |  |  |

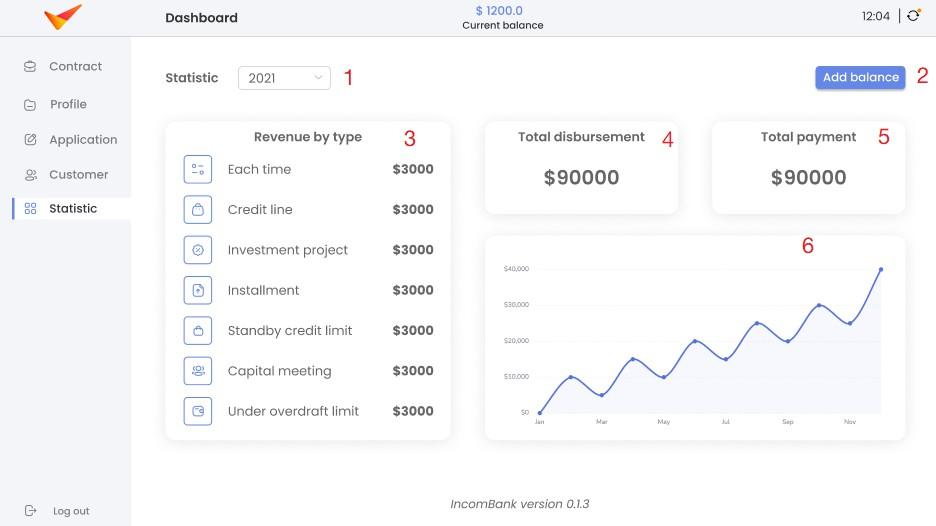
## Edit Customer Info screen



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| N  o | Name | Type | Feature | Calling Conditio  n | Not e |
| 1 | name\_EditText | EditTe  xt | Enter input as name | Type in  edit text |  |
| 2 | phoneNumber\_EditText | EditTe xt | Enter input as phone number | Type in edit text |  |

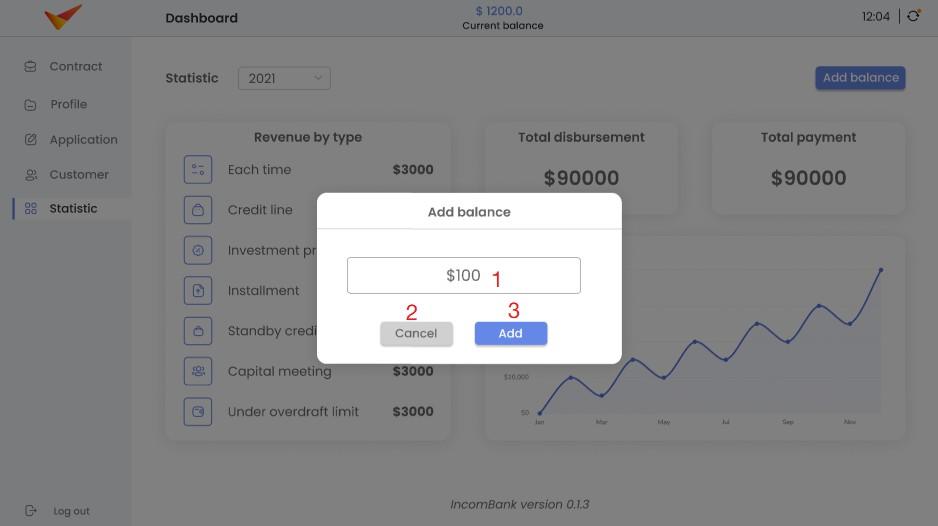
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 3 | email\_EditText | EditTe xt | Enter input as email | Type in edit text |  |
| 4 | dateOfBirth\_EditText | EditTe xt | Enter input as date of birth | Type in edit text |  |
| 5 | identityNumber\_EditText | EditTe xt | Enter input as identity number | Type in edit text |  |
| 6 | identityCardCreatedDate\_Edit Text | EditTe xt | Enter input as identityCardCreatedD  ate | Type in edit text |  |
| 7 | address\_EditText | EditTe xt | Enter input as address | Type in edit text |  |
| 8 | permanentResidence\_EditText | EditTe xt | Enter input as permanentResidence | Type in edit text |  |

## Admin screen



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| N  o | Name | Type | Feature | Calling  Conditio n | Not e |
| 1 | year\_Spinner | Spinner | Get data  correspondin g to year | Choose  item in spinner |  |
| 2 | addBalance\_Button | Button | Add balance | Click the button |  |
| 3 | revenueByType\_RecyclerVie w | RecyclerVie w | Show data |  |  |
| 4 | totalDisburse\_CardView | CardView | Show data | Click the button |  |
| 5 | totalPayment\_CardView | CardView | Show data | Click the button |  |
| 6 | Chart | Chart | Show data |  |  |

## Add balance



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No | Name | Type | Feature | Calling Condition | Note |
| 1 | amount\_EditText | EditText | Enter input as  amount | Type in edit text |  |
| 2 | cancel\_Button | Button | Navigate one back | Click the button |  |
| 3 | add\_Button | Button | Add balance | Click the button |  |

# Summary

## Development Environment and Deployment Environment

## Development Environment

* + - Operating system: Windows 10, MacOS.
    - Database: MongoDB.
    - Analysis, design tools: Figma, draw.io, notion.
    - IDE: Android Studio, Visual Studio Code.
    - Support tools: Postman, MongoDBCompass.
    - Frameworks used: Android/Kotlin, ExpressJS.

## Deployment Environment

* + - Operating system: Android 10 or above.
    - Ram: 2GB
    - Screen size: 1920x1080px.

## Achievement

* Meet the system requirements, functional requirements of the project.
* Gorgeous User Interface, Extremely good User Experience, ease to use.
* Follow Android best practices, such as Hilt dependencies injection, ViewModel, data binding.
* Clean code, easy to maintain or upgrade.
* Apply object-oriented design, analysis, programming.
* Use websocket to synchronize data between devices.
* Statistics function helps the bank track their business performance.

## Future development

* Apply Machine Learning/Deep Learning to get data from image (indentity card)
* Apply Blockchain for creating smart contract
* Create event/discount for customer (based on customer’s point/level)
* Intergrate calculating salary for staff (based on clock in/clock out function)
* Intergrate bar code scanner to get customer infor (based on barcode in customer info page)