Variable Names

Name	Type	Structure	Visibility	Purpose
contractOwner	address	N/A	public	store the owner of the contract
User	struct	`{bytes32 credId, address[] ethAddress,bytes32 brightID,bytes32 bloomId,bytes32 passportScanId}`	N/A	Structure to store user id details
credId	mapping	`(address ethAddress => bytes32 credId)`	N/A	mapping of user address with the credId
userInfo	mapping	`(bytes32 credId => User user)`	N/A	mapping of credId with the User struct
credIdList	bytes32[]	N/A	N/A	Array of all the credIds of the users. **Note: In future, implement this off-chain**
activeLoans	mapping	`(bytes32 credId => bytes32 loanId)`	N/A	mapping of credId with the active loanId as user has One Loan request. (Active Loans means: Requested/ Approved/ Accepeted/ Funded loans)
inactiveLoans	mapping	`(bytes32 credId => bytes32[] loanId)`	N/A	mapping of credId with inactive loanIds. (Inactive loans means: Not approved/ Not Accepted/ Not funded loans)
LoanTerms	struct	`{uint256 principalAmt, uint8 interestRate, uint8 duration}`	N/A	Structure to store requested loan details. (principalAmt: amount in DAI, interestRate: rate is calculated annually, duration: duration is in months)
loanRequested	mapping	`(bytes32 loanId => LoanTerms loanTerms)`	N/A	mapping of loanId with the loan terms requested (loanTerms struct) with the user address

Name	Type	Structure	Visibility	Purpose
loanApproved	mapping	`(bytes32 loanId => LoanTerms loanTerms)`	N/A	mapping of loan terms approved (loanTermsApproved struct) with the user address
loanStatus	mapping	`(bytes32 loanId => uint8 loanStatus)`	N/A	Store the loan status. `[0: No status, 1: Requested, 2: Approved, 3: Not Approved, 4: Accepted, 5: Not Accepted, 6: Funded, 7: Not Funded]`
loanRequestedDate	mapping	`(bytes32 loanId => uint256 loanRequesteddate)`	N/A	Store date and time when loan is requested
loanApproveDate	mapping	`(bytes32 loanId => uint256 oanApprovedDate)`	N/A	Store date and time when loan is approved or not approved
loanAcceptedDate	mapping	`(bytes32 loanId) => uint256 loanAcceptedDate)`	N/A	Store date and time when loan approved terms is accepted or not accepted by user
loanFundedDate	mapping	`(bytes32 loanId => uint256 loanFundedDate)`	N/A	Store date and time when loan is funded or not funded to user
PaymentDetails	struct	`{uint256[] dueDates, uint8[] statuses, uint256 monthlyPayment, uint256 totalPayment}`	N/A	track payment details of the loan. **Note: In fututre, may get rid of uint256 totalPayment variable.** `[0: Pending, 1: Successful, 2: Missed]`
userLoanPayment	mapping	`(bytes32 loanId => PaymentDetails paymentDetails)`	N/A	mapping of loanId with the PaymentDetails struct

Data connections

```
ethAddress --> credID --> user details
                  |-----> loan id --> loan status
```

```
                |---------> request/approved loan data
                |---------> funded loan data
```

Functions

Function name	Input Parameters	Return Parameters	Called By	Description
isOwner	N/A	`bool`	User/ Owner	To check the contract owner address
registerUser	N/A	`bytes32 credId`	User/Owner	To register the user logged in for the first time and creates the credId for new users registered.
getCredId	N/A	`bytes32 credId`	Logged in User	Get the cred id for the current logged in user
getAllCredId	N/A	`bytes32[] credIdList`	Only Owner	Get all the cred ids for all users
getUser	`bytes32 credId`	`address[] ethAddress, bytes32 brightId, bytes32 bloomId, bytes32 passportScanId` **(User struct)**	Logged in User/ Owner	Get user details (Ids)
getActiveLoan	`bytes32 credId`	`bytes32 loanId` **(activeLoans[credId])**	Logged in User/ Owner	Get the loanId of the active loan
getInactiveLoans	`bytes32 credId`	`bytes32[] loanId`	Logged in User/ Owner	Get the inactive loans from the credId of the user

Function name	Input Parameters	Return Parameters	Called By	Description
requestLoan	`uint256 principalAmt, uint8 interestRate, uint8 duration`	`bytes32 loanId, uint8 loanStatus, uint256 loanRequestDate`	Logged In User	submit the loan request and store the loan requested terms
approveLoan	`bytes32 loanId, bool yesNo`	`uint8 loanStatus, uint256 loanApproveDate`	Only Owner	approve the loan request raised by user
acceptLoan	`byes32 loanId, bool yesNo`	`uint8 loanStatus, uint256 loanAcceptedDate`	Only Logged In User	accept the approved loan terms
fundLoan	`byes32 loanId, bool yesNo`	`uint8 loanStatus`**,** **(**`uint256[] dueDates, uint8 statuses, uint256 monthlyPayment, uint256 totalPayment`**(PaymentDetails (struct)))** `,uint256 loanFundedDate`	Only Owner	fund the user with the approved loan amount, create the payment schedule and change loan status to funded
getLoanStatus	`bytes32 loanId`	`uint8 loanStatus`	Logged In User/ Owner	Get the status of the loan
getRequestedLoan	`bytes32 loanId`	`uint256 principalAmt, uint8 interestRate, uint8 duration`**(loanRequested (mapping (bytes32 loanId => LoanTerms loanTerms))),**`uint256 loanRequestedDate, uint8 loanStatus`	Logged in User/ Owner	Get the loan requested terms

Function name	Input Parameters	Return Parameters	Called By	Description
getApprovedLoan	`bytes32 loanId`	`uint256 principalAmt, uint8 interestRate, uint8 duration`**(loanApproved (mapping (bytes32 loanId => LoanTerms loanTerms))),**`uint256 loanApprovedDate, uint8 loanStatus`	Logged in User/ Owner	Get the loan details for the approved loan terms
getPayment	`bytes32 loanId`	`PaymentDetails (struct)`	Owner	Get the payment schedule and monthly payment and total payment details
getPayment	N/A	`uint256[] dueDates, uint8 statuses, uint256 monthlyPayment, uint256 totalPayment` **(PaymentDetails (struct))`**	Only Logged in User	Get the payment schedule and monthly payment and total payment details using user address
makePayment	N/A	N/A	Only logged in User	Make the payment for the monthly loan amount