DESIGN DOCUMENT

Hubba - Account Recovery

Version 1.6
Prepared By: Development Hell
Class: CECS 491B-05
Date: February 16, 2023

Github Repository:
https://github.com/DevelopmentHellaHell/SeniorProject

Team Leader
Kevin Dinh
Members
Garrett Tsumaki
Bryan Tran
Jett Sonoda
Tien Nguyen
Darius Koroni

POC: Jett.Sonoda@student.csulb.edu

Revision History

Version	Overview	Date
1.0	Requirements Established	January 23, 2023
1.1	Initial HLD	January 25, 2023
1.2	Ideal Case HLD Done	January 26, 2023
1.3	Successful LLD Done	January 29, 2023
1.4	Successful LLD Revamped	January 30, 2023
1.5	Failure LLD Done	February 1, 2023
1.6	Relation Table Updates	February 16, 2023

Table of Contents

Overview

This document is intended to provide all need-to-know sources of the design of the Account Recovery feature for potential cases of fixing, improving, debugging, and understanding all components of this feature. This document contains multiple abstraction levels of design, including use cases and database tables.

Requirements to Satisfy

Taken directly from either the BRD or Client's Core Component Requirements email.

Authentication

Requirements

- Account is locked until a valid account recovery mechanism is performed by the account owner or by the system admin. Upon successful account recovery, the failed authentication attempts resets.

Account Recovery

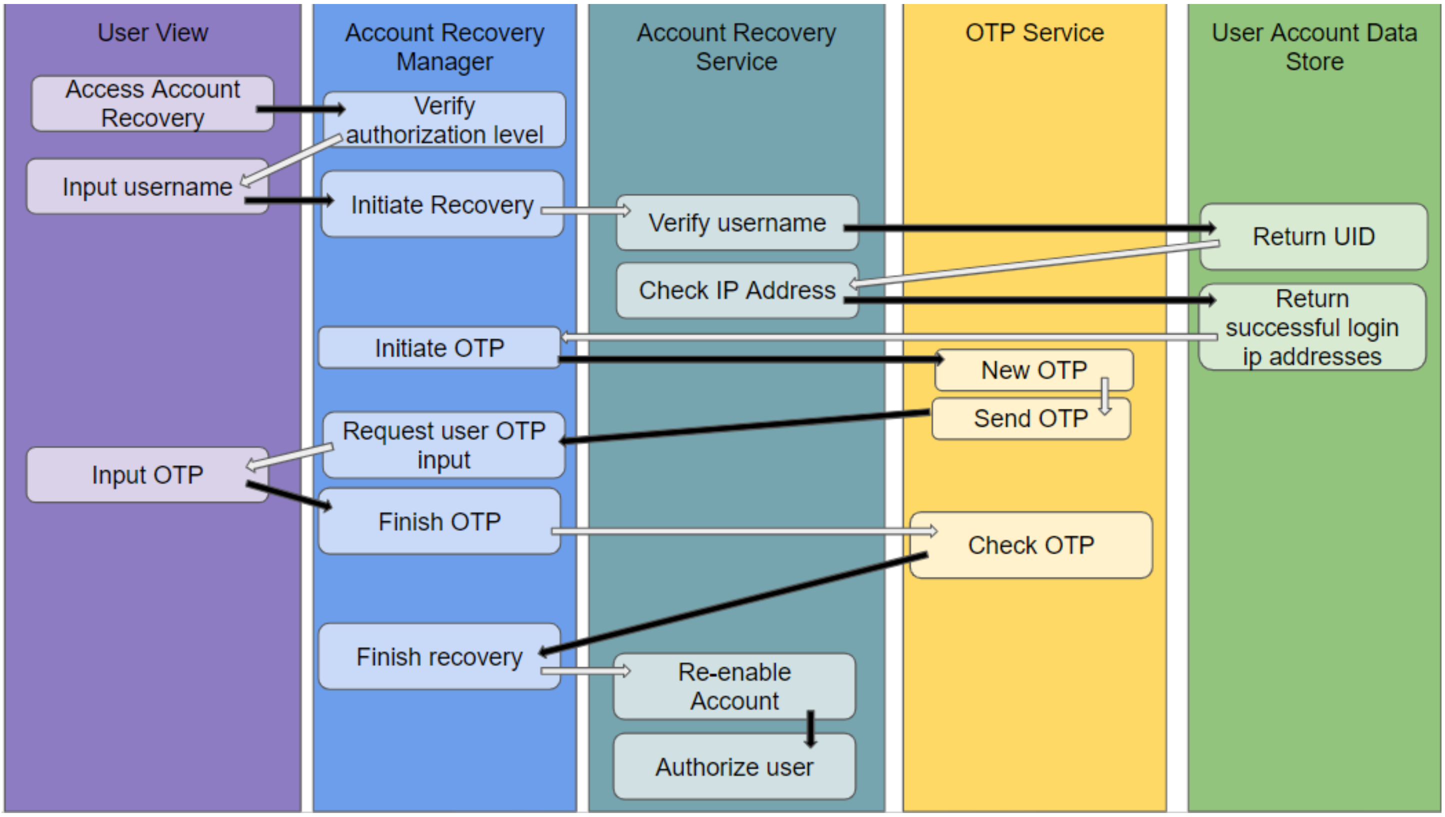
Requirements:

- The user must provide assigned username and valid OTP to submit account recovery request
- An authorized system admin will be able to view the latest account recovery requests for all non-admin users.
- Upon successful account recovery by authorized system admin, the user will be able to authenticate into the system.
- System failures from this feature must not result in the system going offline

Use Cases:

- ○ Pre-conditions
  1. User must not have an active authenticated session on the device, otherwise the user is unable to perform the operation
  2. User must be on account recovery view
- ○ Success
  1. User provides assigned username and valid OTP. Request is made available to authorized system admin users within 5 seconds. A system message displays "Account recovery request sent" within 5 seconds of invocation of request.
  2. An authorized system admin completes account recovery for user. A system message displays "Account recovery completed successfully for user" within 5 seconds of invocation. Affected user regains access to the system within 5 seconds of invocation.
- ○ Failure Cases
  1. User provides invalid username. A system message displays "Invalid username or OTP provided. Retry again or contact system administrator"
  2. User provides valid username, but invalid OTP. A system message displays "Invalid username or OTP provided. Retry again or contact system administrator"
  3. User provides valid username and valid OTP. Request is not available to authorized system admin users.
  4. User provides valid username and valid OTP. Request is available to authorized system admin users. System message does not display within 5 seconds on invocation.
  5. An authorized system admin completes account recovery for user. System message does not display within 5 seconds on invocation.
  6. An authorized system admin completes account recovery for user. System message does display within 5 seconds on invocation. Affected user does not regain access.
  7. An authorized system admin completes account recovery for user. System message does display within 5 seconds on invocation. Affected user does not regain access within 5 seconds.

High-Level Design

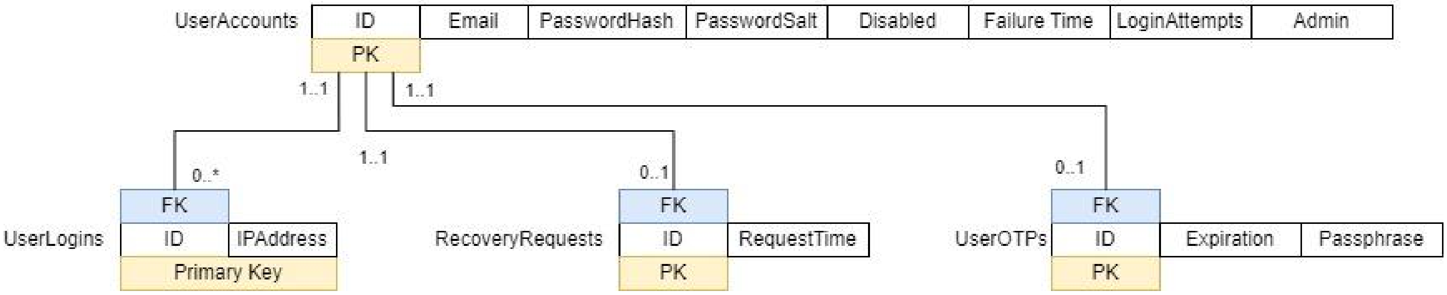


NOTE: This is in the case of the user (non-admin) attempting account recovery while their IP address matches one of the successful login IP addresses currently stored. No human intervention is needed in this case.

🟨 Account Recovery Design

Low-Level Design

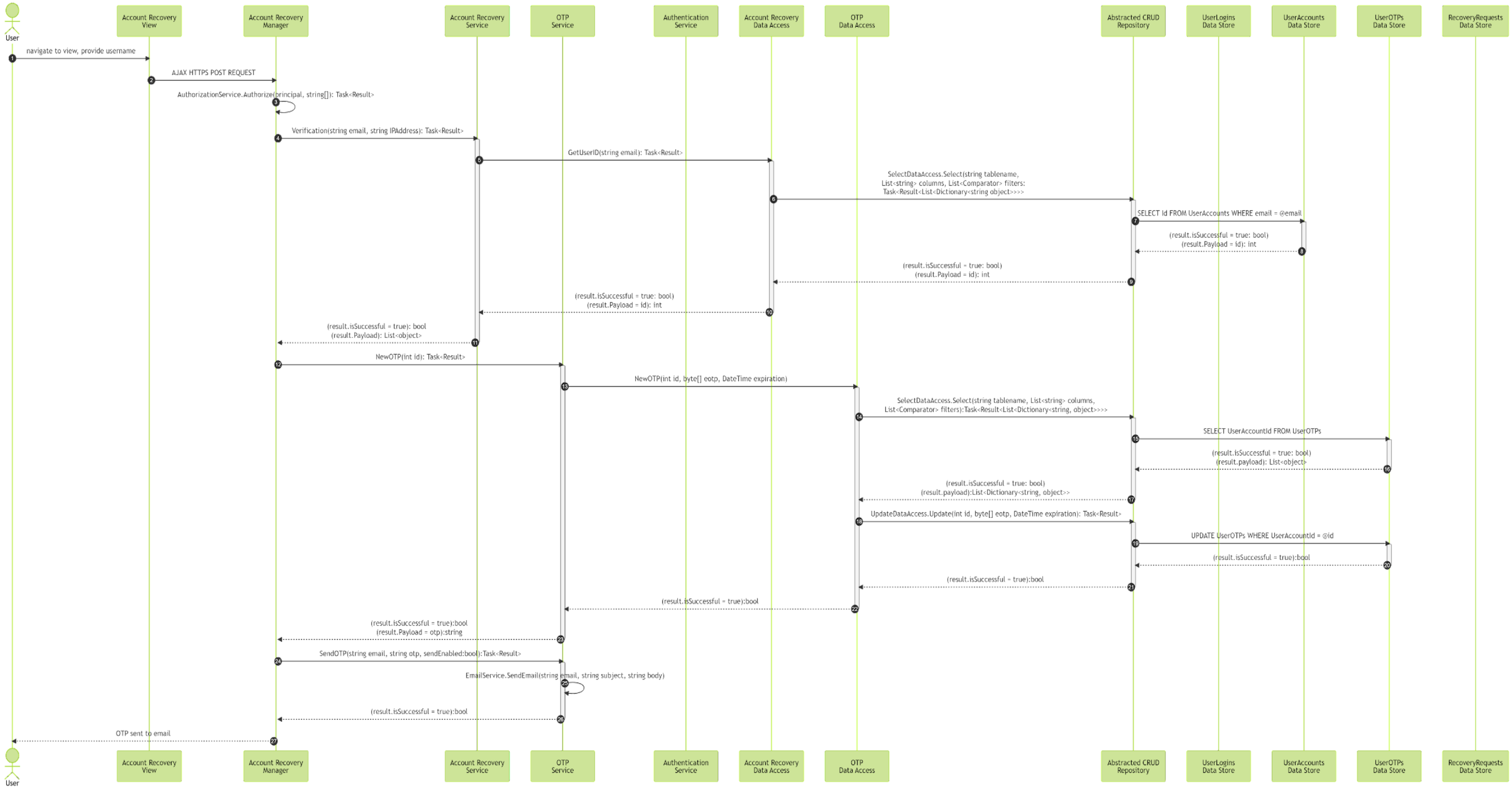
Relational Table(s)



Successful Use Case(s)

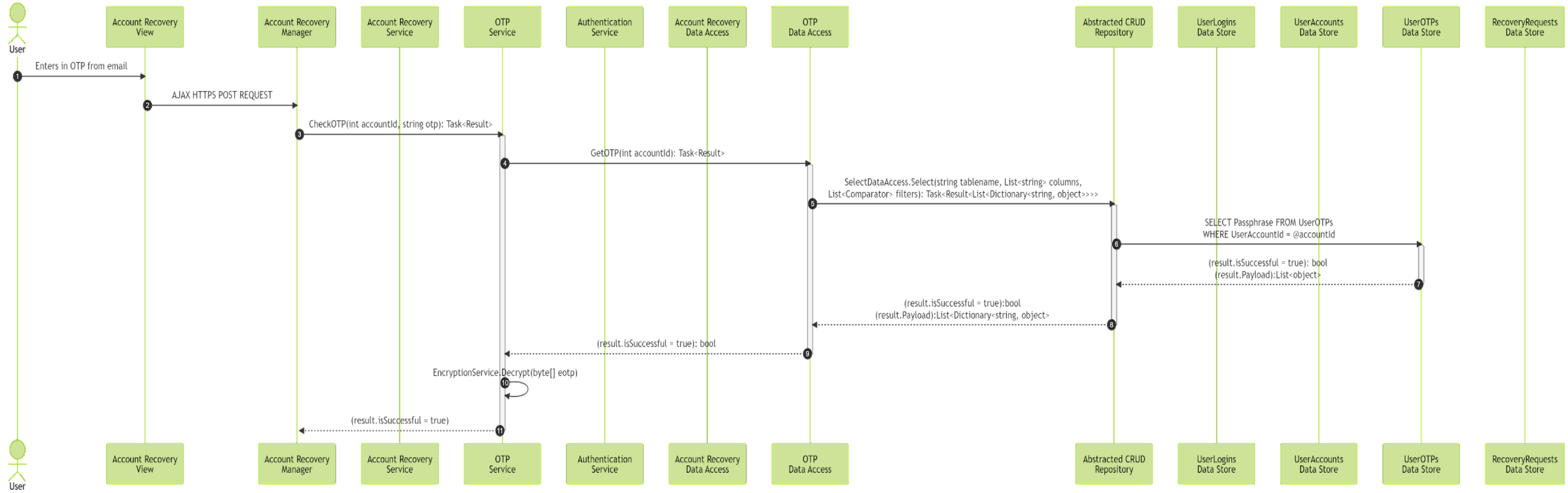
The successful use case is broken into three parts due to the restrictions of Google Doc and readability. The last step of each part 1 then directly goes to the first step of part 2. Similarly, the last step of part 2 then directly goes to the first step of part 3.

<u>Part 1 of 3:</u>

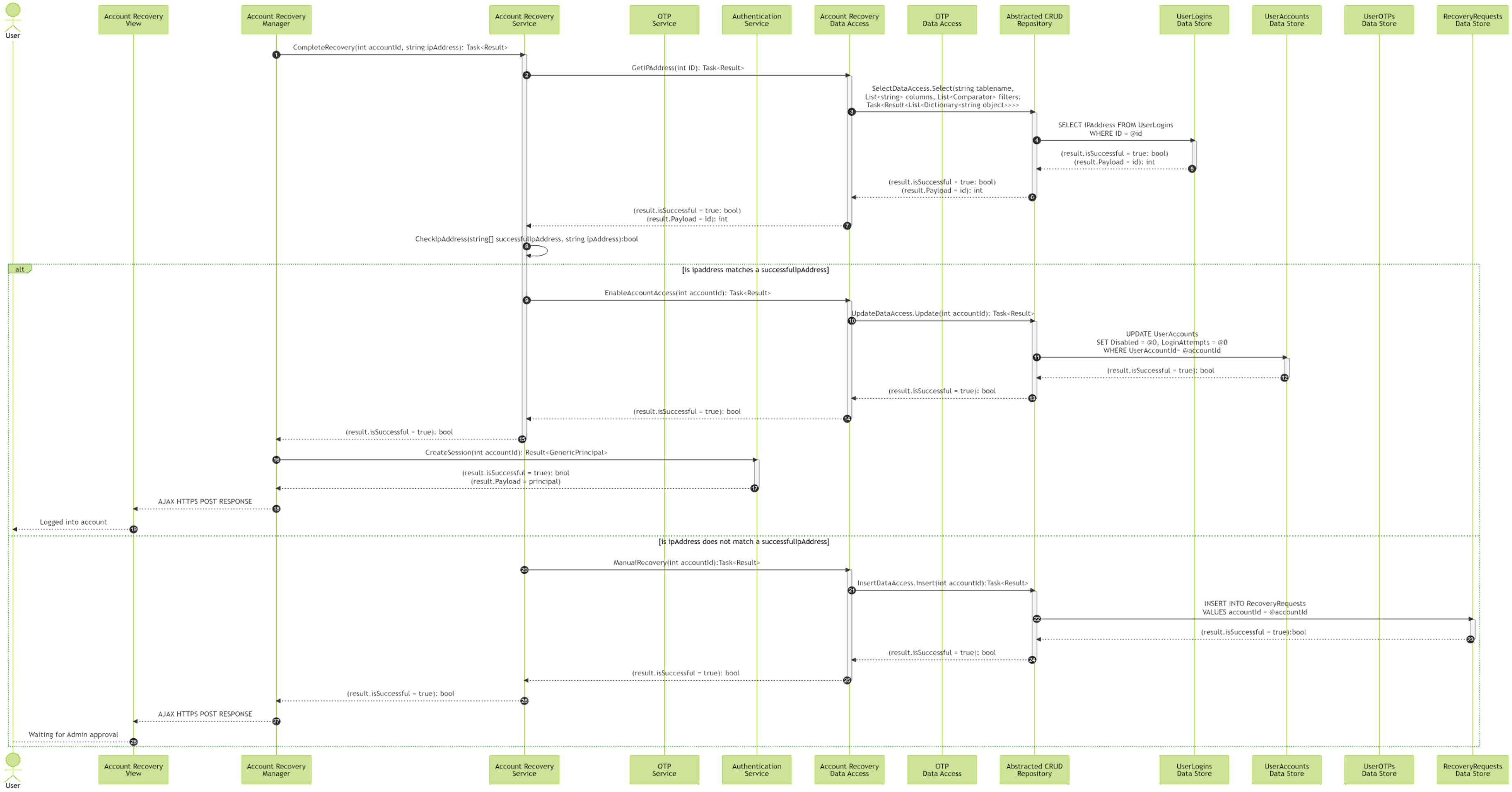


User
Account Recovery View
Account Recovery Manager
Account Recovery Service
OTP Service
Authentication Service
Account Recovery Data Access
OTP Data Access
Abstracted CRUD Repository
UserLogins Data Store
UserAccounts Data Store
UserOTPs Data Store
RecoveryRequests Data Store
1 navigate to view, provide username
2 AJAX HTTPS POST REQUEST
3 AuthorizationService.Authorize(principal, string[]): Task<Result>
4 Verification(string email, string IPAddress): Task<Result>
5 GetUserID(string email): Task<Result>
6 SelectDataAccess.Select(string tablename, List<string> columns, List<Comparator> filters: Task<Result<List<Dictionary<string object>>>
7 SELECT Id FROM UserAccounts WHERE email = @email
8 (result.isSuccessful = true: bool) (result.Payload = id): int
9 (result.isSuccessful = true: bool) (result.Payload = id): int
10 (result.isSuccessful = true: bool) (result.Payload = id): int
11 (result.isSuccessful = true): bool (result.Payload): List<object>
12 NewOTP(int id): Task<Result>
13 NewOTP(int id, byte[] eotp, DateTime expiration)
14 SelectDataAccess.Select(string tablename, List<string> columns, List<Comparator> filters):Task<Result<List<Dictionary<string, object>>>
15 SELECT UserAccountId FROM UserOTPs
16 (result.isSuccessful = true: bool) (result.payload): List<object>
17 (result.isSuccessful = true: bool) (result.payload):List<Dictionary<string, object>>
18 UpdateDataAccess.Update(int id, byte[] eotp, DateTime expiration): Task<Result>
19 UPDATE UserOTPs WHERE UserAccountId = @id
20 (result.isSuccessful = true):bool
21 (result.isSuccessful = true):bool
22 (result.isSuccessful = true):bool
23 (result.isSuccessful = true):bool (result.Payload = otp):string
24 SendOTP(string email, string otp, sendEnabled:bool):Task<Result>
25 EmailService.SendEmail(string email, string subject, string body)
26 (result.isSuccessful = true):bool
27 OTP sent to email
User
Account Recovery View
Account Recovery Manager
Account Recovery Service
OTP Service
Authentication Service
Account Recovery Data Access
OTP Data Access
Abstracted CRUD Repository
UserLogins Data Store
UserAccounts Data Store
UserOTPs Data Store
RecoveryRequests Data Store

Part 2 of 3:


User
Account Recovery View
Account Recovery Manager
Account Recovery Service
OTP Service
Authentication Service
Account Recovery Data Access
OTP Data Access
Abstracted CRUD Repository
UserLogins Data Store
UserAccounts Data Store
UserOTPs Data Store
RecoveryRequests Data Store
1 Enters in OTP from email
2 AJAX HTTPS POST REQUEST
3 CheckOTP(int accountId, string otp): Task<Result>
4 GetOTP(int accountId): Task<Result>
5 SelectDataAccess.Select(string tablename, List<string> columns, List<Comparator> filters): Task<Result<List<Dictionary<string, object>>>>
6 SELECT Passphrase FROM UserOTPs WHERE UserAccountId = @accountId
7 (result.isSuccessful = true): bool (result.Payload):List<object>
8 (result.isSuccessful = true):bool (result.Payload):List<Dictionary<string, object>
9 (result.isSuccessful = true): bool
10 EncryptionService.Decrypt(byte[] eotp)
11 (result.isSuccessful = true)
User
Account Recovery View
Account Recovery Manager
Account Recovery Service
OTP Service
Authentication Service
Account Recovery Data Access
OTP Data Access
Abstracted CRUD Repository
UserLogins Data Store
UserAccounts Data Store
UserOTPs Data Store
RecoveryRequests Data Store

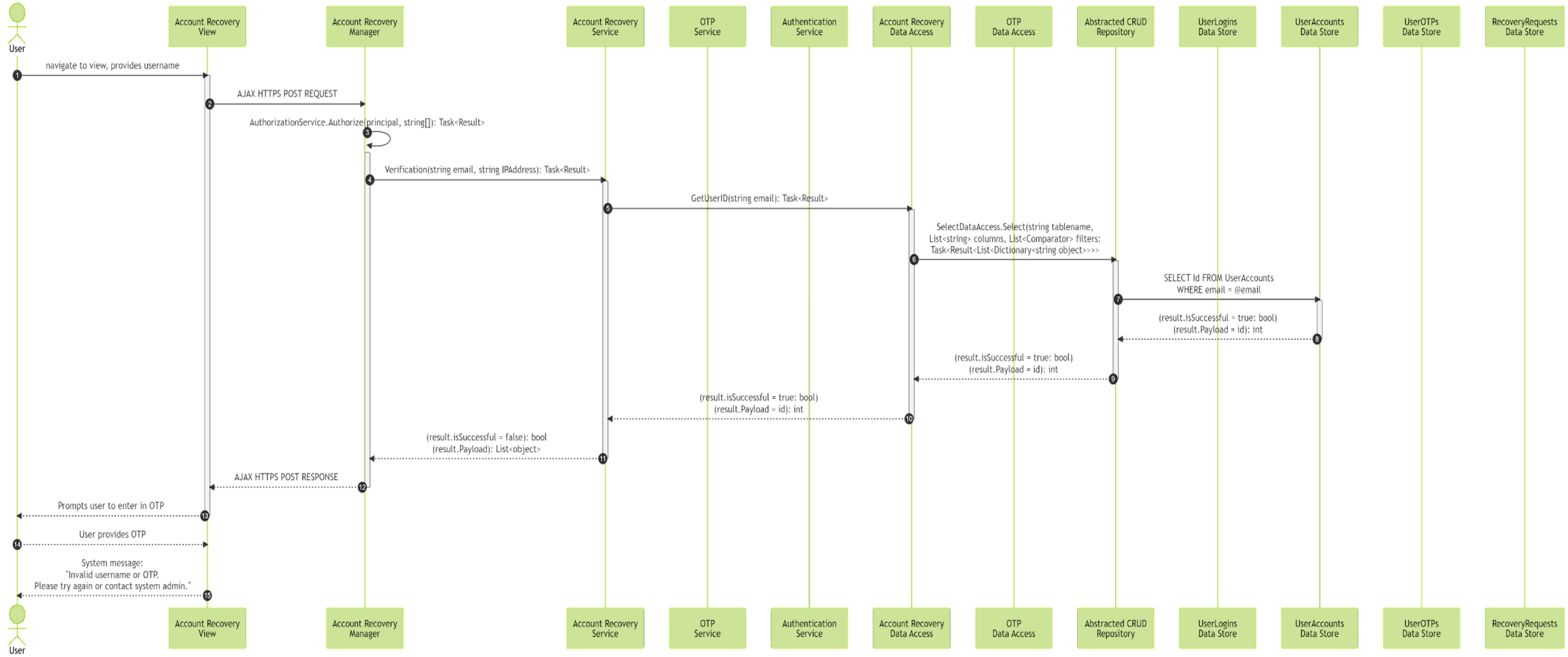
Part 3 of 3:


User
Account Recovery View
Account Recovery Manager
Account Recovery Service
OTP Service
Authentication Service
Account Recovery Data Access
OTP Data Access
Abstracted CRUD Repository
UserLogins Data Store
UserAccounts Data Store
UserOTPs Data Store
RecoveryRequests Data Store
1 CompleteRecovery(int accountId, string ipAddress): Task<Result>
2 GetIPAddress(int ID): Task<Result>
3 SelectDataAccess.Select(string tablename, List<string> columns, List<Comparator> filters: Task<Result<List<Dictionary<string object>>>>
4 SELECT IPAddress FROM UserLogins WHERE ID = @id
5 (result.isSuccessful = true: bool) (result.Payload = id): int
6 (result.isSuccessful = true: bool) (result.Payload = id): int
7 (result.isSuccessful = true: bool) (result.Payload = id): int
8 CheckIpAddress(string[] successfulIpAddress, string ipAddress):bool
alt
[is ipaddress matches a successfullpAddress]
9 EnableAccountAccess(int accountId): Task<Result>
10 UpdateDataAccess.Update(int accountId): Task<Result>
11 UPDATE UserAccounts SET Disabled = @0, LoginAttempts = @0 WHERE UserAccountId= @accountId
12 (result.isSuccessful = true): bool
13 (result.isSuccessful = true): bool
14 (result.isSuccessful = true): bool
15 (result.isSuccessful = true): bool
16 CreateSession(int accountId): Result<GenericPrincipal>
17 (result.isSuccessful = true): bool (result.Payload = principal)
18 AJAX HTTPS POST RESPONSE
19 Logged into account
[is ipAddress does not match a successfullpAddress]
20 ManualRecovery(int accountId):Task<Result>
21 InsertDataAccess.Insert(int accountId):Task<Result>
22 INSERT INTO RecoveryRequests VALUES accountId = @accountId
23 (result.isSuccessful = true):bool
24 (result.isSuccessful = true): bool
25 (result.isSuccessful = true): bool
26 (result.isSuccessful = true): bool
27 AJAX HTTPS POST RESPONSE
28 Waiting for Admin approval
User
Account Recovery View
Account Recovery Manager
Account Recovery Service
OTP Service
Authentication Service
Account Recovery Data Access
OTP Data Access
Abstracted CRUD Repository
UserLogins Data Store
UserAccounts Data Store
UserOTPs Data Store
RecoveryRequests Data Store

Failure Use Case(s)

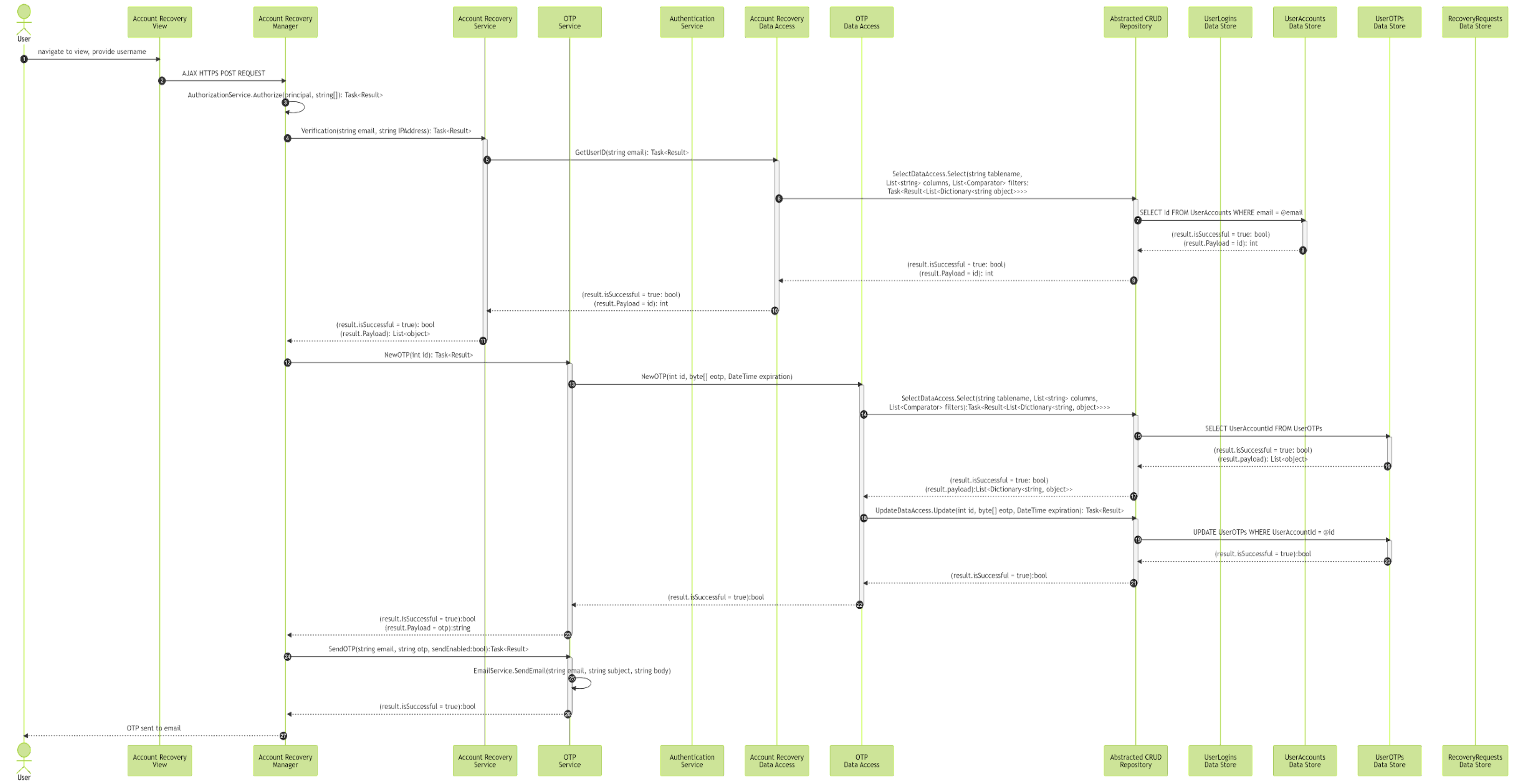
1. User provides invalid username (fix result to false)

<u>Part 1 of 2:</u>

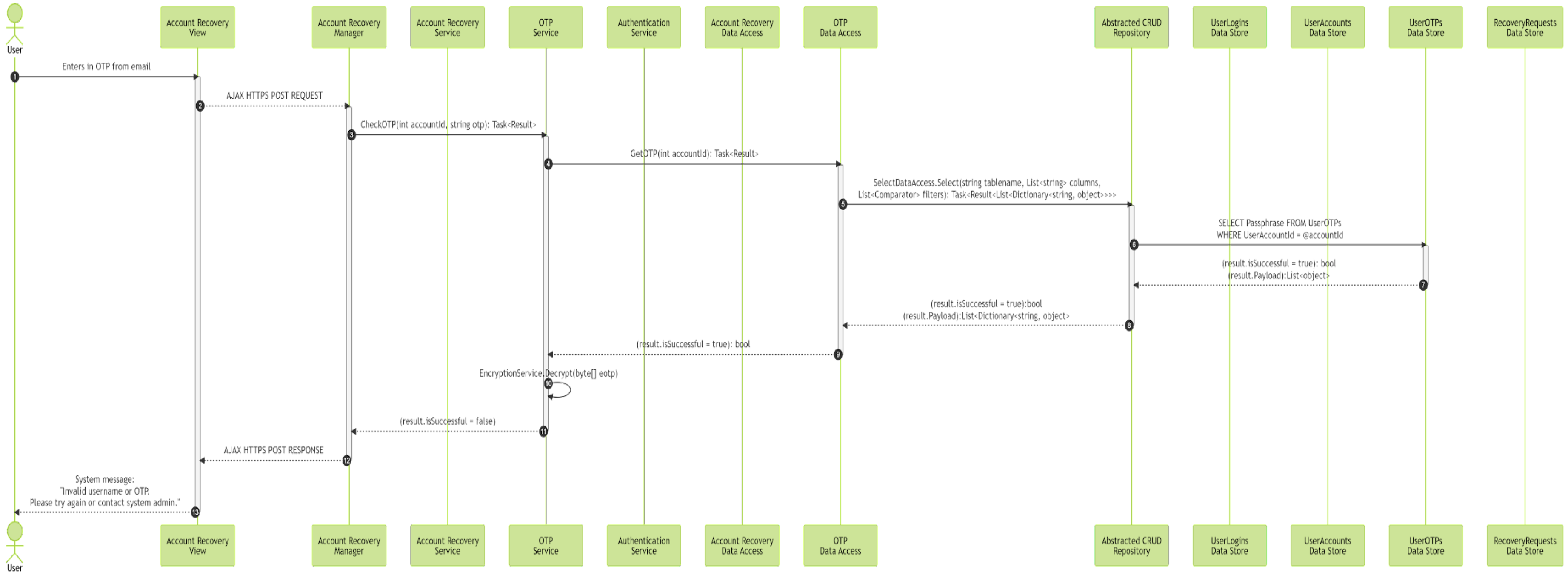

User
Account Recovery View
Account Recovery Manager
Account Recovery Service
OTP Service
Authentication Service
Account Recovery Data Access
OTP Data Access
Abstracted CRUD Repository
UserLogins Data Store
UserAccounts Data Store
UserOTPs Data Store
RecoveryRequests Data Store
1 navigate to view, provides username
2 AJAX HTTPS POST REQUEST
3 AuthorizationService.Authorize(principal, string[]): Task<Result>
4 Verification(string email, string IPAddress): Task<Result>
5 GetUserID(string email): Task<Result>
6 SelectDataAccess.Select(string tablename, List<string> columns, List<Comparator> filters: Task<Result<List<Dictionary<string object>>>>
7 SELECT Id FROM UserAccounts WHERE email = @email
8 (result.isSuccessful = true: bool) (result.Payload = id): int
9 (result.isSuccessful = true: bool) (result.Payload = id): int
10 (result.isSuccessful = true: bool) (result.Payload = id): int
11 (result.isSuccessful = false): bool (result.Payload): List<object>
12 AJAX HTTPS POST RESPONSE
13 Prompts user to enter in OTP
14 User provides OTP
15 System message: "Invalid username or OTP. Please try again or contact system admin."
User
Account Recovery View
Account Recovery Manager
Account Recovery Service
OTP Service
Authentication Service
Account Recovery Data Access
OTP Data Access
Abstracted CRUD Repository
UserLogins Data Store
UserAccounts Data Store
UserOTPs Data Store
RecoveryRequests Data Store

2. User provides invalid OTP

Part 1 of 2:


User
Account Recovery View
Account Recovery Manager
Account Recovery Service
OTP Service
Authentication Service
Account Recovery Data Access
OTP Data Access
Abstracted CRUD Repository
UserLogins Data Store
UserAccounts Data Store
UserOTPs Data Store
RecoveryRequests Data Store
1 navigate to view, provide username
2 AJAX HTTPS POST REQUEST
3 AuthorizationService.Authorize(principal, string[]): Task<Result>
4 Verification(string email, string IPAddress): Task<Result>
5 GetUserID(string email): Task<Result>
6 SelectDataAccess.Select(string tablename, List<string> columns, List<Comparator> filters: Task<Result<List<Dictionary<string object>>>>
7 SELECT Id FROM UserAccounts WHERE email = @email
8 (result.isSuccessful = true: bool) (result.Payload = id): int
9 (result.isSuccessful = true: bool) (result.Payload = id): int
10 (result.isSuccessful = true: bool) (result.Payload = id): int
11 (result.isSuccessful = true): bool (result.Payload): List<object>
12 NewOTP(int id): Task<Result>
13 NewOTP(int id, byte[] eotp, DateTime expiration)
14 SelectDataAccess.Select(string tablename, List<string> columns, List<Comparator> filters):Task<Result<List<Dictionary<string, object>>>>
15 SELECT UserAccountId FROM UserOTPs
16 (result.isSuccessful = true: bool) (result.payload): List<object>
17 (result.isSuccessful = true: bool) (result.payload):List<Dictionary<string, object>>
18 UpdateDataAccess.Update(int id, byte[] eotp, DateTime expiration): Task<Result>
19 UPDATE UserOTPs WHERE UserAccountId = @id
20 (result.isSuccessful = true):bool
21 (result.isSuccessful = true):bool
22 (result.isSuccessful = true):bool
23 (result.isSuccessful = true):bool (result.Payload = otp):string
24 SendOTP(string email, string otp, sendEnabled:bool):Task<Result>
25 EmailService.SendEmail(string email, string subject, string body)
26 (result.isSuccessful = true):bool
27 OTP sent to email
User
Account Recovery View
Account Recovery Manager
Account Recovery Service
OTP Service
Authentication Service
Account Recovery Data Access
OTP Data Access
Abstracted CRUD Repository
UserLogins Data Store
UserAccounts Data Store
UserOTPs Data Store
RecoveryRequests Data Store

<u>Part 2 of 2:</u>



User
Account Recovery View
Account Recovery Manager
Account Recovery Service
OTP Service
Authentication Service
Account Recovery Data Access
OTP Data Access
Abstracted CRUD Repository
UserLogins Data Store
UserAccounts Data Store
UserOTPs Data Store
RecoveryRequests Data Store
1
Enters in OTP from email
2
AJAX HTTPS POST REQUEST
3
CheckOTP(int accountId, string otp): Task<Result>
4
GetOTP(int accountId): Task<Result>
5
SelectDataAccess.Select(string tablename, List<string> columns, List<Comparator> filters): Task<Result<List<Dictionary<string, object>>>>
6
SELECT Passphrase FROM UserOTPs WHERE UserAccountId = @accountId
7
(result.isSuccessful = true): bool (result.Payload):List<object>
8
(result.isSuccessful = true):bool (result.Payload):List<Dictionary<string, object>
9
(result.isSuccessful = true): bool
10
EncryptionService.Decrypt(byte[] eotp)
11
(result.isSuccessful = false)
12
AJAX HTTPS POST RESPONSE
13
System message: "Invalid username or OTP. Please try again or contact system admin."
User
Account Recovery View
Account Recovery Manager
Account Recovery Service
OTP Service
Authentication Service
Account Recovery Data Access
OTP Data Access
Abstracted CRUD Repository
UserLogins Data Store
UserAccounts Data Store
UserOTPs Data Store
RecoveryRequests Data Store

References

This document elaborates client's requirements[1] for one of the product's core components - ACCOUNT RECOVERY.

The requirement is designed based on the system architecture provided in High-Level Design Document[2].

The diagrams included in this document utilized diagrams.net[3] and Mermaid.js[4].

[1] Client's email, 10/17/2022

[2] High-Level Design Document, URL: https://github.com/DevelopmentHellaHell/SeniorProject/blob/b43182c4076d471ef03520052675f1f88371dafd/docs/HL%20Design/DevelopmentHell%20HLD%20v1.2.pdf

[3] https://www.diagrams.net/

[4] https://mermaid-js.github.io/mermaid/#/