LOW-LEVEL DESIGNS DOCUMENT

Hubba - Registration

Version 2.0
Prepared By: Development Hell
Class: CECS 491-04
Date: November 20, 2022

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

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

Revision History

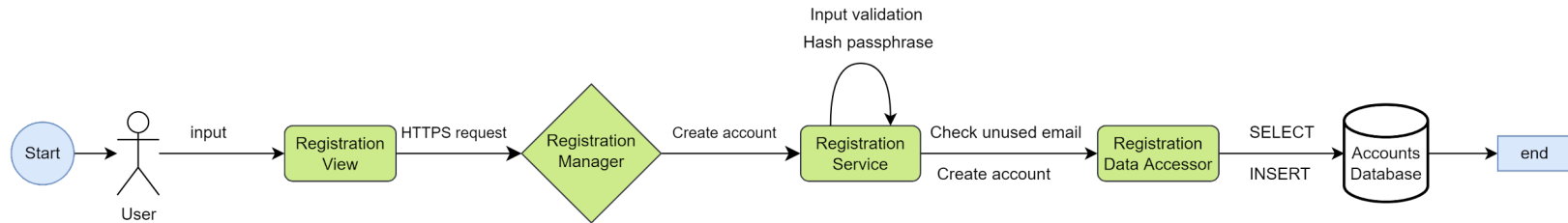| Version	Overview	Date
1.0	Initial LLD	10/21/2022
1.1	Final Test Plans	10/31/2022
1.2	Diagram revisions	11/7/2022
2.0	Revision after Milestone 2 Delivery	11/20/2022

Table of Contents

High-Level Design

1. Activity flow - Client request:

2. Activity flow - System response:
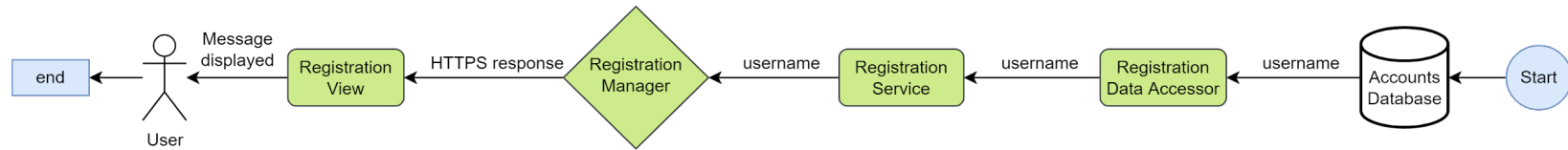
Low-Level Design

User story: As a user, I can create a new account with a valid email and a valid passphrase.

Preconditions:
1. User must have no active authenticated session.
2. User must be on the Registration view.

Business Rules:
- User provides a valid email and a valid passphrase to register.
  - A valid email is in format: email@example.com
  - A valid passphrase satisfies the following constraints:
    i. Minimum length of 8 characters
    ii. Consists of the following:
      - blank space
      - a-z
      - A-Z
      - 0-9
      - . , @ ! -

Class Diagram

Result

+bool: IsSuccessful
+string: ErrorMessage
+object?: Payload

+Result(bool: IsSuccessful, string: ErrorMessage, object?: Payload)

Account

+string: username
+string: passphraseHash
+string: passphraseSalt
+string: displayName
+int: id
+bool: adminAccount
+DateTime: birthDate
+DateTime: lastLogIn
+string: email

RegistrationManager

+createAccount(string: jsonString) : Task

RegistrationService

-static string: connectionString
-RegistrationDataAccess: _registrationDAO

+BirthdateValidation(DateTime: Birthdate) : Result
+EmailValidation(string: Email) : Result
+PassphraseValidation(string: Passphrase) : Result
+ObjectToDictionary(object: obj) : Result

«interface»
ILogger

+Log(LogLevel: level, string: userName, string: message) : Task<Result>

Logger

-readonly ILoggerDataAccess _dataAccess
-readonly Category _category

+Logger(ILoggerDataAccess: dataAccess, Category: category)

«interface»
ILoggerDataAccess

+LogData(LogLevel: logLevel, Category: category, string: userName, string: message) : Task<Result>
+SelectLogs(List<string<: columns, Dictionary>string, object>: filters) : Task<Result>

RegistrationDataAccess

-InsertDataAccess: _insertDataAccess
-SelectDataAccess: _selectDataAccess
-static string: _tableName

+LoggerDataAccess(string: connectionString) : Result

LoggerDataAccess

-InsertDataAccess: _insertDataAccess
-SelectDataAccess: _selectDataAccess
-static string: _tableName

+LoggerDataAccess(string: connectionString) : Result

SelectDataAccess

-string: connectionPath

-SendQuery(SqlCommand: query) : Task<Result>
+Select(string: table, List<string<: columns, Dictionary>string, object>: filters) : Task<Result>

InsertDataAccess

-string: connectionPath

-SendQuery(SqlCommand: query) : Task<Result>
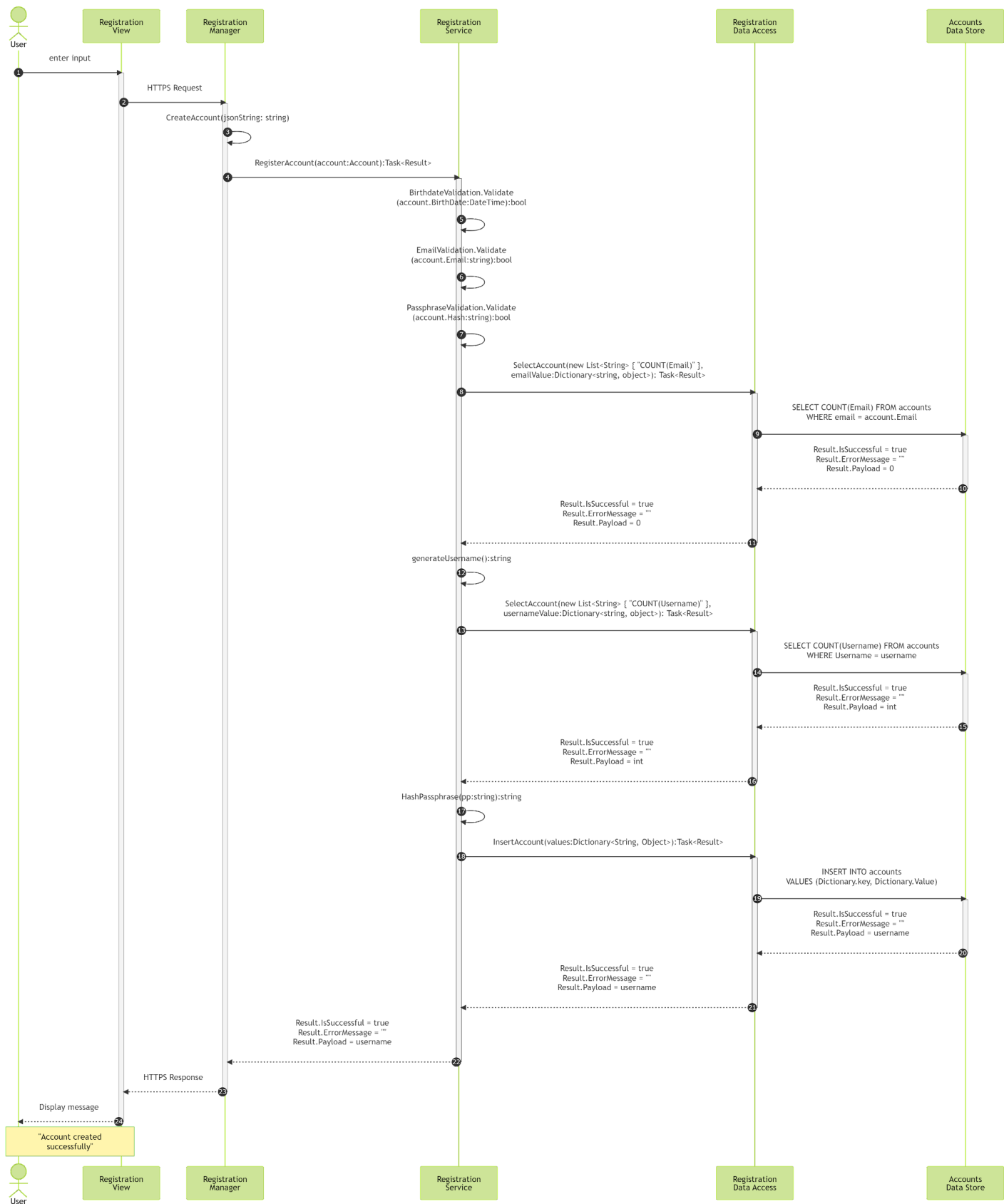+Insert(string: table, Dictionary<string, object>: values) : Task<Result>

5

Successful Use Case(s)

1. User provides a valid email and a valid passphrase. Within 5 seconds, the system assigns a system-wide unique username then displays a message to the user - "Account created successfully".

   Note:
   - Email character validation requirement will be satisfied in step 6
   - Email uniqueness validation requirement will be satisfied in steps 8-11
   - Passphrase character validation requirement will be satisfied in step 7
   - Username generation requirement will be satisfied in steps 12-16
   - Account creation requirement will be satisfied in steps 18-21
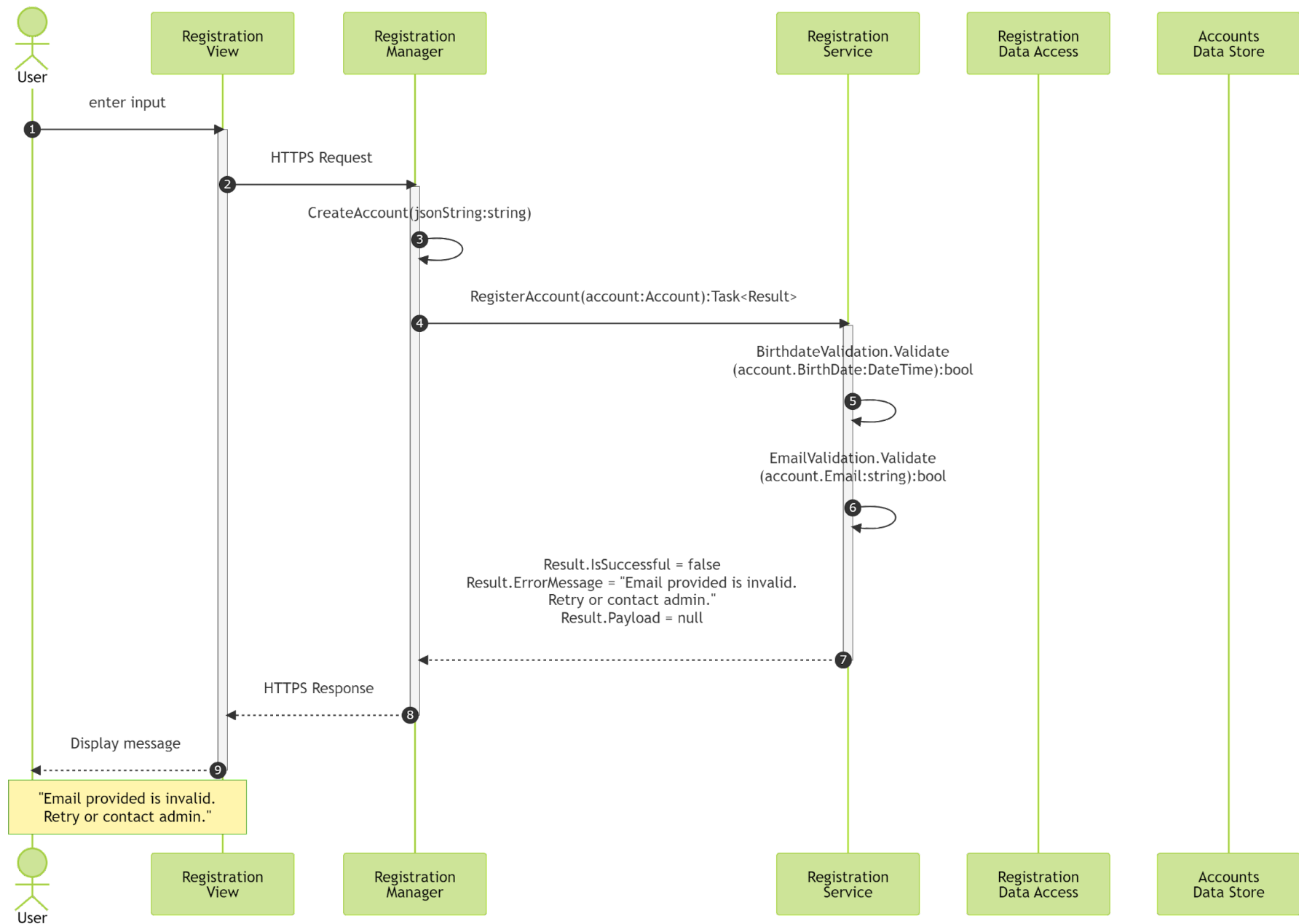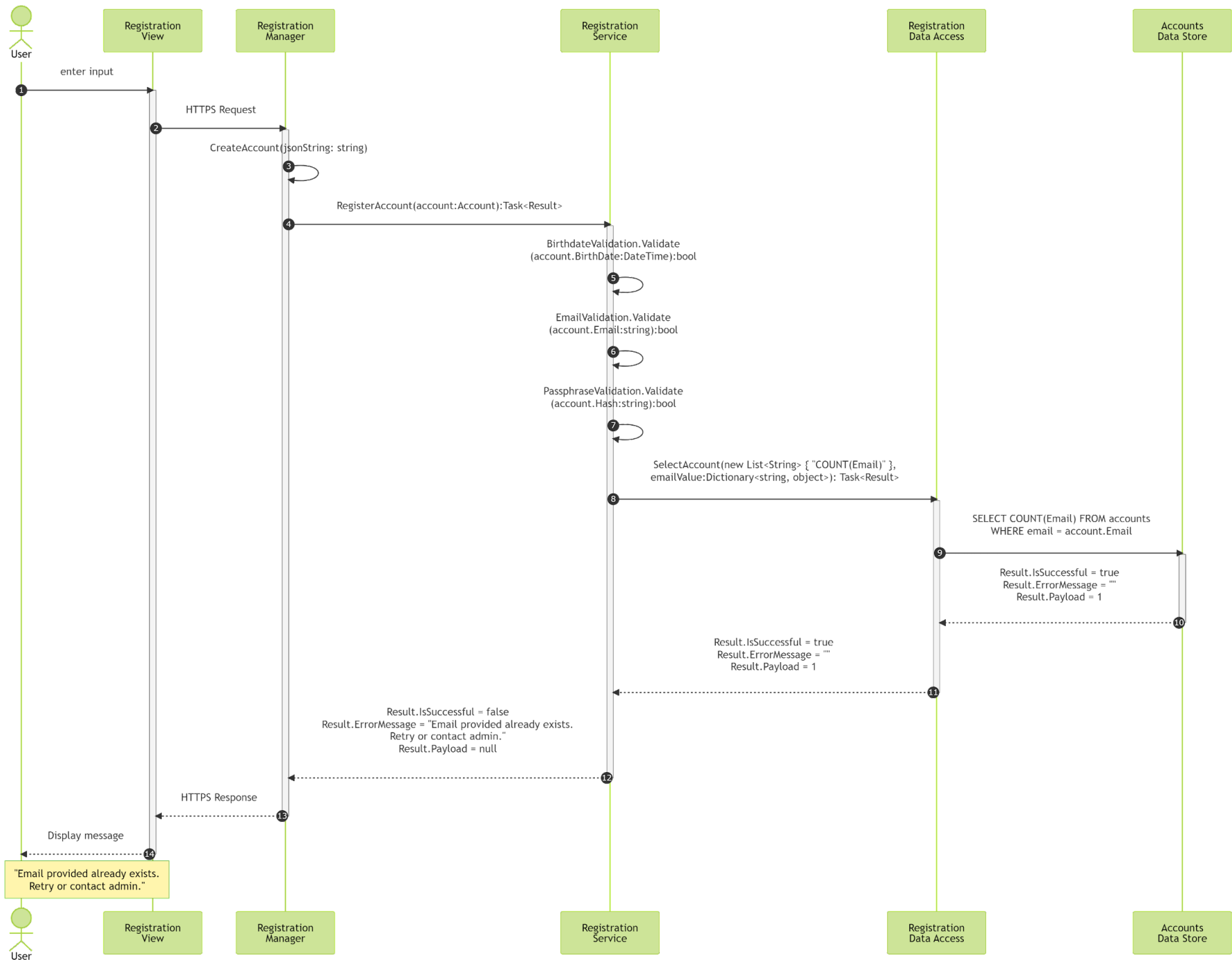   - Message display requirement will be satisfied in step 24

User

Registration
View

Registration
Manager

Registration
Service

Registration
Data Access

Accounts
Data Store

enter input

1

HTTPS Request

2

CreateAccount(jsonString: string)

3

RegisterAccount(account:Account):Task<Result>

4

BirthdateValidation.Validate
(account.BirthDate:DateTime):bool

5

EmailValidation.Validate
(account.Email:string):bool

6

PassphraseValidation.Validate
(account.Hash:string):bool

7

SelectAccount(new List<String> ["COUNT(Email)"],
emailValue:Dictionary<string, object>): Task<Result>

8

SELECT COUNT(Email) FROM accounts
WHERE email = account.Email

9

Result.IsSuccessful = true
Result.ErrorMessage = ""
Result.Payload = 0

10

Result.IsSuccessful = true
Result.ErrorMessage = ""
Result.Payload = 0

11

generateUsername():string

12

SelectAccount(new List<String> { "COUNT(Username)" },
usernameValue:Dictionary<string, object>): Task<Result>

13

SELECT COUNT(Username) FROM accounts
WHERE Username = username

14

Result.IsSuccessful = true
Result.ErrorMessage = ""
Result.Payload = int

15

Result.IsSuccessful = true
Result.ErrorMessage = ""
Result.Payload = int

16

HashPassphrase(pp:string):string

17

InsertAccount(values:Dictionary<String, Object>):Task<Result>

18

INSERT INTO accounts
VALUES (Dictionary.key, Dictionary.Value)

19

Result.IsSuccessful = true
Result.ErrorMessage = ""
Result.Payload = username

20

Result.IsSuccessful = true
Result.ErrorMessage = ""
Result.Payload = username

21

Result.IsSuccessful = true
Result.ErrorMessage = ""
Result.Payload = username

22

HTTPS Response

23

Display message

24

"Account created
successfully"

7

Failure Use Case(s)

2. An invalid email is provided on registration. The system displays a message "Invalid email provided. Retry again or contact system administrator". The system doesn't create a new account.
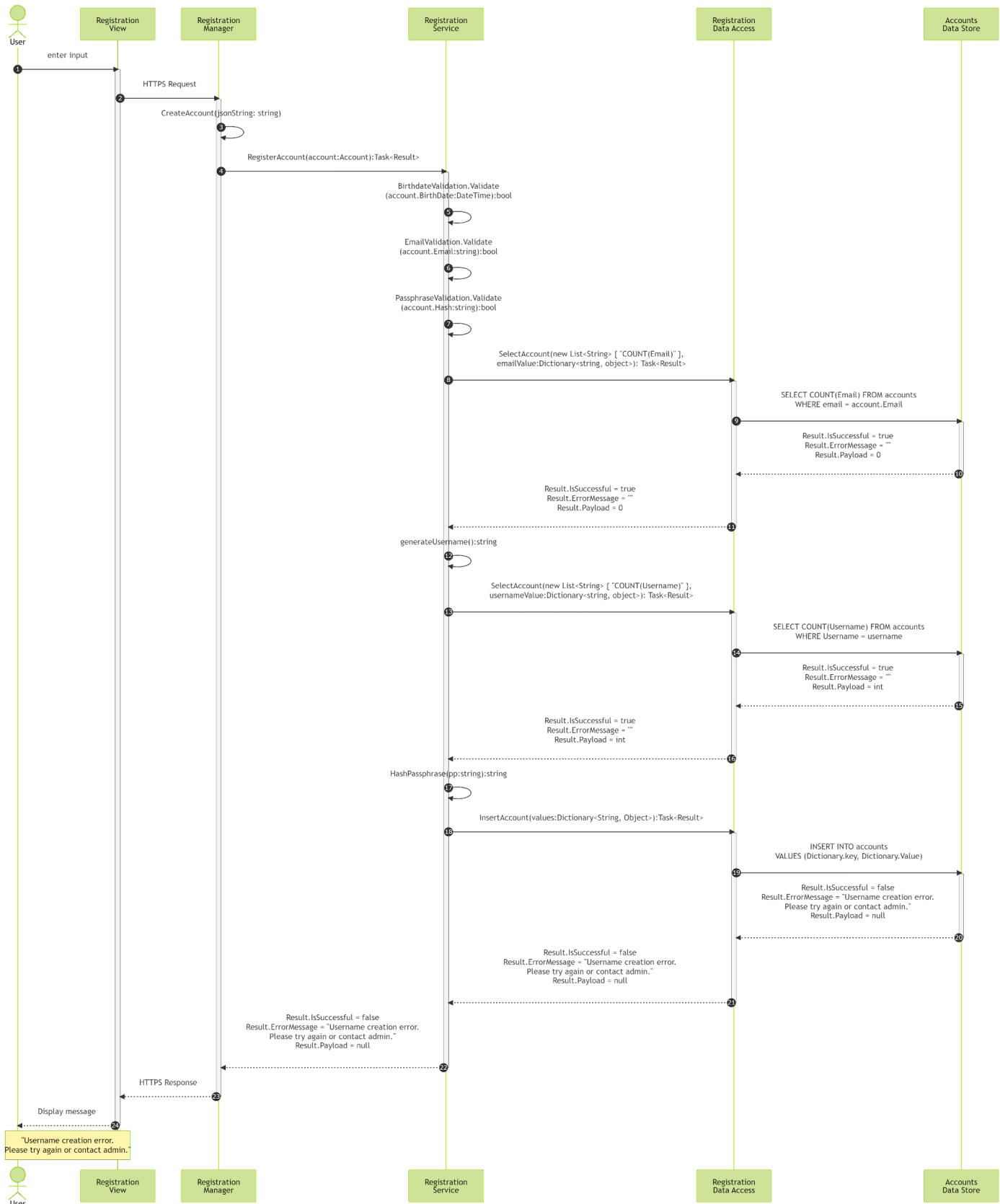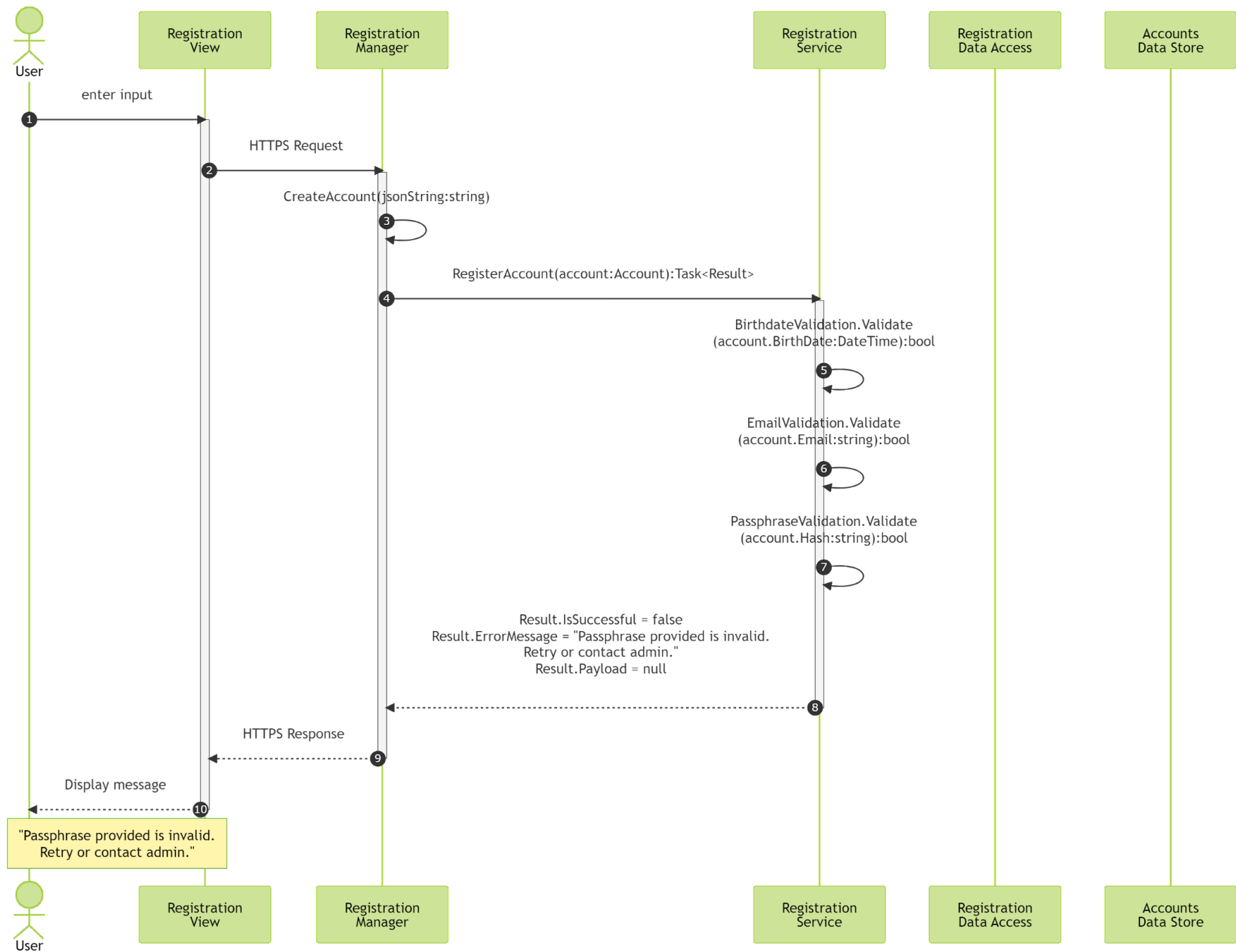
   2a. Email's format is not satisfied

User

Registration View

Registration Manager

Registration Service

Registration Data Access

Accounts Data Store

enter input

① ②

HTTPS Request

CreateAccount(jsonString:string)

③

RegisterAccount(account:Account):Task<Result>

④

BirthdateValidation.Validate (account.BirthDate:DateTime):bool

⑤

EmailValidation.Validate (account.Email:string):bool

⑥

Result.IsSuccessful = false
Result.ErrorMessage = "Email provided is invalid. Retry or contact admin."
Result.Payload = null

⑦

HTTPS Response

⑧

Display message

⑨

"Email provided is invalid. Retry or contact admin."

2b. Email provided already exists

3. User provides a valid email and a valid passphrase to register; however, the system doesn't assign a system-wide username. User receives a message - "Username creation error. Please try again or contact system administrators.". There's no account created.
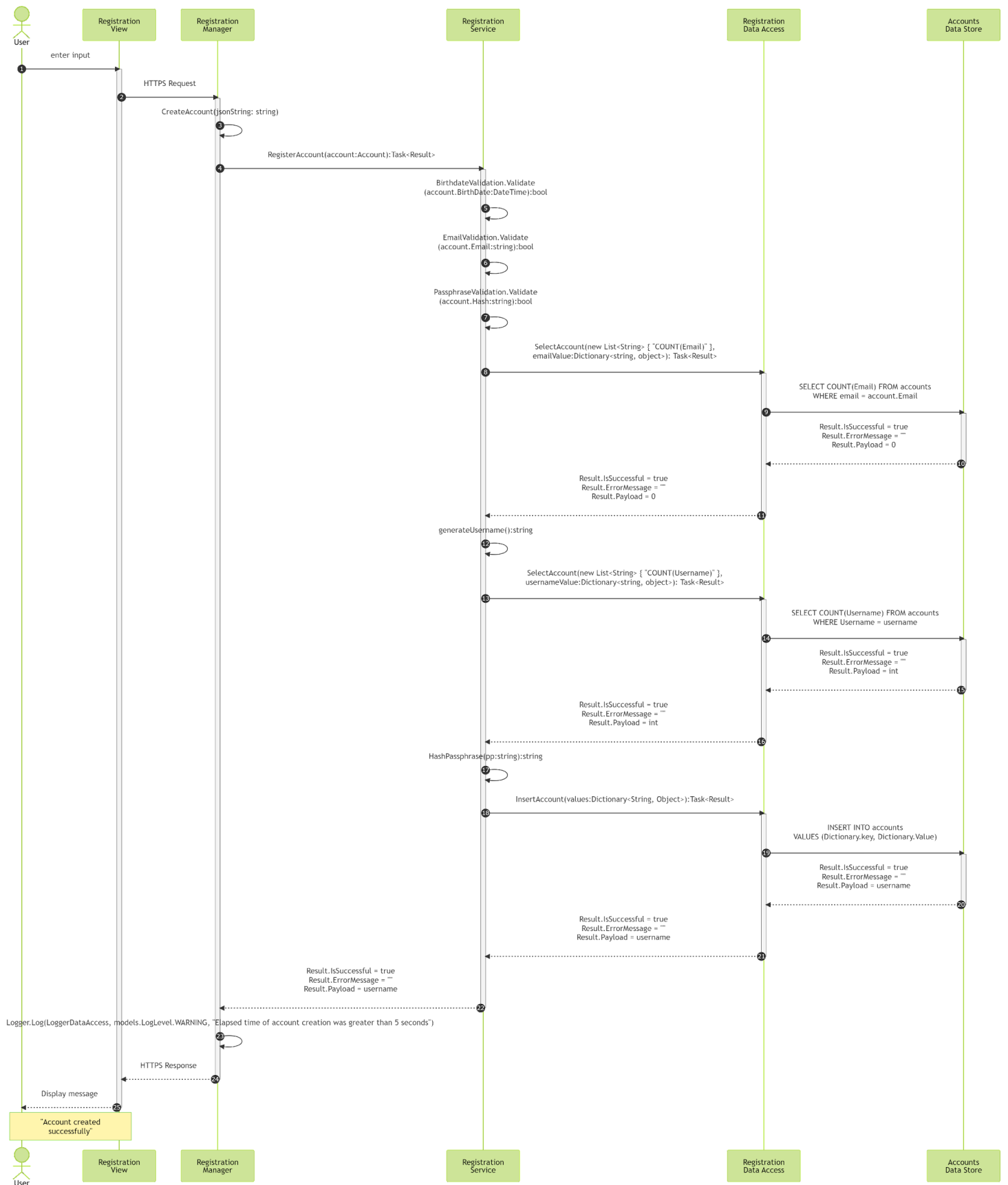
User

Registration View

Registration Manager

Registration Service

Registration Data Access

Accounts Data Store

enter input

1

HTTPS Request

2

CreateAccount(jsonString: string)

3

RegisterAccount(account:Account):Task<Result>

4

BirthdateValidation.Validate
(account.BirthDate:DateTime):bool

5

EmailValidation.Validate
(account.Email:string):bool

6

PassphraseValidation.Validate
(account.Hash:string):bool

7

SelectAccount(new List<String> { "COUNT(Email)" },
emailValue:Dictionary<string, object>): Task<Result>

8

SELECT COUNT(Email) FROM accounts
WHERE email = account.Email

9

Result.IsSuccessful = true
Result.ErrorMessage = ""
Result.Payload = 0

10

Result.IsSuccessful = true
Result.ErrorMessage = ""
Result.Payload = 0

11

generateUsername():string

12

SelectAccount(new List<String> { "COUNT(Username)" },
usernameValue:Dictionary<string, object>): Task<Result>

13

SELECT COUNT(Username) FROM accounts
WHERE Username = username

14

Result.IsSuccessful = true
Result.ErrorMessage = ""
Result.Payload = int

15

Result.IsSuccessful = true
Result.ErrorMessage = ""
Result.Payload = int

16

HashPassphrase(pp:string):string

17

InsertAccount(values:Dictionary<String, Object>):Task<Result>

18

INSERT INTO accounts
VALUES (Dictionary.key, Dictionary.Value)

19

Result.IsSuccessful = false
Result.ErrorMessage = "Username creation error.
Please try again or contact admin."
Result.Payload = null

20

Result.IsSuccessful = false
Result.ErrorMessage = "Username creation error.
Please try again or contact admin."
Result.Payload = null

21

Result.IsSuccessful = false
Result.ErrorMessage = "Username creation error.
Please try again or contact admin."
Result.Payload = null

22

HTTPS Response

23

Display message

24

"Username creation error.
Please try again or contact admin."

13

4. User provides an invalid passphrase to register. User receives a message - "Passphrase provided is invalid. Retry or contact system administrator". No account is created.

5. User registers with a valid email and valid passphrase. The system was able to assign a system-wide username. Entire process took longer than 5 seconds. A system log entry is recorded. Account is created.

Integration Test Plan[1]

ID	Description	Extra Information	Input	Expected Output
Reg01	Successfully create the 1st account	Follow standard AAA testing guidelines	Email=TestEmail1@gmail.com Passphrase=Test Case Reg-01 Birthdate =2001-01-01	Unique Username
Reg02	Successfully create the 2nd account with multiple periods after the "@" sign	Follow standard AAA testing guidelines	Email=TestEmail2@gmail.com Passphrase=Test Case Reg-02 Birthdate =2001-01-01	Unique Username
Reg03	Attempt to create the 3rd account with invalid email of no "@" sign	Follow standard AAA testing guidelines	Email=Test3.com Passphrase=Test Case Reg-03 Birthdate =2001-01-01	System Message: "Email is invalid. Retry or contact admin."
Reg04	Attempt to create the 4th account with invalid email of no period after the "@" sign	Follow standard AAA testing guidelines	Email=Test4@com Passphrase=Test Case Reg-04 Birthdate =2001-01-01	System Message: "Email is invalid. Retry or contact admin."
Reg05	Attempt to create the 5th account with invalid email of no arguments after the period	Follow standard AAA testing guidelines	Email=Test5@gmail. Passphrase=Test Case Reg-05 Birthdate =2001-01-01	System Message: "Email is invalid. Retry or contact admin."
Reg06	Attempt to create the 6th account with invalid email of no arguments before the period	Follow standard AAA testing guidelines	Email=Test6@.com Passphrase=Test Case Reg-06 Birthdate =2001-01-01	System Message: "Email is invalid. Retry or contact admin."
Reg07	Attempt to create the	Follow standard AAA testing guidelines	Email=TestEmail1@gmail.com	System Message: "Email

---

[1] https://www.guru99.com/test-case.html

	7th account with duplicate email		Passphrase=Test Case Reg-07 Birthdate =2001-01-01	provided already exists. Retry or contact admin."
Reg08	Attempt to create the 8th account with invalid passphrase of no invalid characters	Follow standard AAA testing guidelines	Email=TestEmail8@gmail.com Passphrase=Test Case Reg(08) Birthdate =2001-01-01	System Message: "Passphrase provided is invalid. Retry or contact admin."
Reg09	Attempt to create the 9th account with invalid passphrase of too little characters	Follow standard AAA testing guidelines	Email=TestEmail9@gmail.com Passphrase=2smol-9 Birthdate =2001-01-01	System Message: "Passphrase provided is invalid. Retry or contact admin."
Reg10	Attempt to create the 10th account	Need to time out the system for 5 seconds. Follow standard AAA testing guidelines	Email=TestEmail10@gmail.com Passphrase=Test Case Reg-10 Birthdate =2001-01-01	Unique username
Reg11	Attempt to create the 11th account	Follow standard AAA testing guidelines	Email=TestEmail11@gmail.com Passphrase=Test Case Reg-11 Birthdate =2015-01-01	System Message: "Age requirement not reached."

References

This document elaborates client's requirements[2] for one of the product's core components - REGISTRATION.
The requirement is designed based on the system architecture provided in High-Level Design Document[3].
The diagrams included in this document utilized diagrams.net[4] and Mermaid.js[5].

---

[2] Client's email, 10/17/2020
[3] High-Level Design Document, URL:
https://github.com/DevelopmentHellaHell/SeniorProject/blob/b43182c4076d471ef03520052675f1f88371dafd/docs/HL%20Design/DevelopmentHell%20HLD%20v1.2.pdf
[4] https://www.diagrams.net/
[5] https://mermaid-js.github.io/mermaid/#/