# theCaterpillars

Emily Thomas Kylota Stewart Jared Williams

eathomas1@ualr.edu bxstewart@ualr.edu jxwilliams11@ualr.edu

Juan Perez Minh Dang

japerez@ualr.edu mqdang@ualr.edu

# Assignment 4 – ADM Login-Registration (20 Points)

# Assignment Description

Using the abstract data model (ADM) approach and the hybrid UC format for writing functional requirements, complete ADM and UCs for

Epic 1. RegistrationAndLogin

As a CELDVUser I can obtain an CELDV account and login, so I have secure access.

Here is the start on this assignment we completed in class on Tues 9/12/2023 – with minor corrections and UC 1.1 completed:

ADM

// List of types

CELDVUserType

RegisteredUsers // The set of CELDV login accounts

RegisteredUser

Employees // The set of employees represented in (an installation of) CELDV

Employee

Employers

Employer

EmployedInJob

FormsNewCompany

// Types need for UC development – add in list below

RegistrationRequest

PendingRegistrationRequests

PendingRegistrationRequest

// ADM Details

CELDVUserType: enum[CELDVUser, AdminUser]

RegisteredUsers: set of RegisteredUser // This is the set of valid login credentials – email checked

RegisteredUser: tuple (

emailAddress:

password:

userType: CELDVUserType

)

Employees set of Employee

Employee: tuple (

firstName:

lastName:

employers: set of ref(EmployedInJob)

)

Employers: set of Employer

Employer: tuple (

companyName:

headquartersAddress: // For prototype, just a single string

descendantCompanies: set of ref(FormsNewCompany)

predecessorCompanies: set of ref(FormsNewCompany

hasEmployed: set of ref(EmployedInJob)

)

EmployedInJob: tuple (

theEmployee: ref(Employee)

withCompany: ref(Employer)

jobTitle:

startDate:

endDate:

)

FormsNewCompany: tuple (

parentEmployer: ref(Employer)

childEmployer: ref(Employer)

onDate: // Date the childEmployer started

)

// Additional abstract types added while writing functional requirements

RegistrationRequest: tuple (

submittedEmail:

proposedPassword:

)

PendingRegistrationRequests: set of PendingRegistrationRequest

PendingRegistrationRequest: tuple (

pendingEmail:

proposedPassword:

expireOnDateTime:

// You will need verificationEmailSent: boolean

referenceToPendingRequest: ref(PendingRegistrationRequest)

)

VerificationEmail: tuple(

Message:

verificationURL: CELDVVerificationLink

)

CELDVVerificationLink: tuple (

uRL: // uRL to our application

refToPendingRegistrationRequest:

delete pendingRegistrationRequest and create entry in RegisteredUsers

)

UCs for Epic 1

Epic 1. RegistrationAndLogin

As a CELDVUser I can obtain an CELDV account and login, so I have secure access.

UC 1.1. submitRegistrationRequest

As a CELDVUser I can register for CELDV login credentials so I can securely login to the CELDV application.

Requires: (registrationRequest, RegisteredUsers, pendingRegistrationRequests)

Case 1: If (submittedEmail does not match an entry in RegisteredUsers

AND submittedEmail does not match an entry in pendingRegistrationRequests), then

registrationRequest has been used to create an entry in pendingRegistrationRequests with correct expireOnDateTime.

Case 2: If (submittedEmail does match an entry in pendingRegistrationRequests)

AND (submittedEmail does not match an entry in RegisteredUsers), then

User has been notified they have a pendingRegistrationRequest.

Case 3: If (submittedEmail does not match an entry in pendingRegistrationRequests)

AND (submittedEmail does match an entry in RegisteredUsers), then

(User has been notified they already have a login account).

// Are we going to include a changePassword UC?

(Not at this time).

There is no Case 4 with submittedEmail matching entries in both pendingRegistrationRequests and registeredUsers. This follows since this UC will not allow duplicate entries in pendingRegistrationRequests.

UC 1.2 sendVerificationEmail

As CELDV I can send a verificationEmail with a verificationEmailLink for pendingRegistrationRequest so I can verify submittedEmail.

/\*

Suggestion

As CELDV I can send a VerificationEmail to verify the submittedEmail in a pendingRegistrationRequest.

Requires: (pendingRegistrationRequest, verificationEmai)

Case 1: If (pendingRegistrationRequests is not empty

AND there is a pending request for which verificationEmailSent is false), then

A verificationEmail has been sent to pendingEmail

AND the verificationURL contains a ref(pendingRegistrationRequest).

// Other cases will specify when “there is noting to do”.

\*/

Requires (pendingRegistrationRequest, verificationEmailLink)

Case 1: If (verificationEmailLink is clicked) AND

(today's date is less than pendingRegistrationRequest expiredOnDateTime) then create new RegisteredUser from pendingEmail and proposedPassword and delete pendingRegistrationRequest.

Case 2: If (verificationEmailLink is clicked) AND

Today's date is greater than pendingRegistrationRequest expiredOnDateTime then the user is sent a new VerificationEmail.

UC 1.3 registeredUserLogin

As CELDVUser I can login with my RegisteredUser account to access the CELDV application.

Requires (RegisteredUsers, RegisteredUser:emailAddress, RegisteredUser:password)

Case 1: If (submittedEmail is not associated with any RegisteredUsers) then

display text “Login Failed” and clear entry boxes.

Case 2: If (submittedPassword is not associated with RegisteredUser:emailAddress) then

display text “Login Failed” and clear entry boxes.

# Due Time/Date

This is a team assignment to be completed no later than

**Sun 9/17/2023 no later than 10:00pm**

Solutions received after the due date and time will be worth a maximum of 8 points.

Please use the following file naming convention:

CPSC 4392 Fall 2023.4.ADM-UCs RegistrationAndLogin.<team name>.<file extension~~>~~[[1]](#footnote-2)

Also include the team name and list of team members at the top of the submitted solution.

1. The client (Dr. Baker) prefers assignments be submitted in an MS Word format (.docx). It is easier to add comments and suggestions using MS Word comments. This is harder to do with PDF files. [↑](#footnote-ref-2)