

Overall module structure

SET BOOL = Check_for_ascending_sorting(array,n)

 If BOOL = true

 BOOL = Check_for_prime_sequence(array,n)

 If BOOL = true

 Set newarray = Check for sub-array summation(array, n, C)

 End-if

 End-if

Procedure Check_for_ascending_sorting(array,n)

For i = 1 to n-1 do:

 If array [i] < array [i-1]

 Return false

 End-if

End-for

Output "Sorted array in ascending order"

Return True

Procedure Check_for_prime_sequence(array,n)

For i = 0 to n-1 do:

 Set count = 0

 If array [i] > 1

 For y = 1 to array[i]

 If remainder of array[i] divide y equal to 0

 count += 1

 end-if

 end-for

 if count >2

 return false

 end-if

 Else if array [i] <= 0

 Return false

 end-if

end-for

Procedure Check for sub-array summation(array, n, C)

For j = 0 to n-1 do:

Set SUM = 0

Set count = 0

While SUM < C :

SUM += array[j + n]

count++

if SUM > C

break the while loop

end-if

if count > (n-1-j)

return false

End-if

if SUM == C

Set new array to store sub-array summation

For z = j to count :

new array[z-j] = array[z]

Return New array

End-for

End-if

End-while

End-for

<<cuhk account>>
- account id: int
- student name :string
- getaccountid() : int
- getstudentname() : string

Link Account
-LinkAccount:string
-password: string
-getLinkAccount:string
-changePassword(): pool

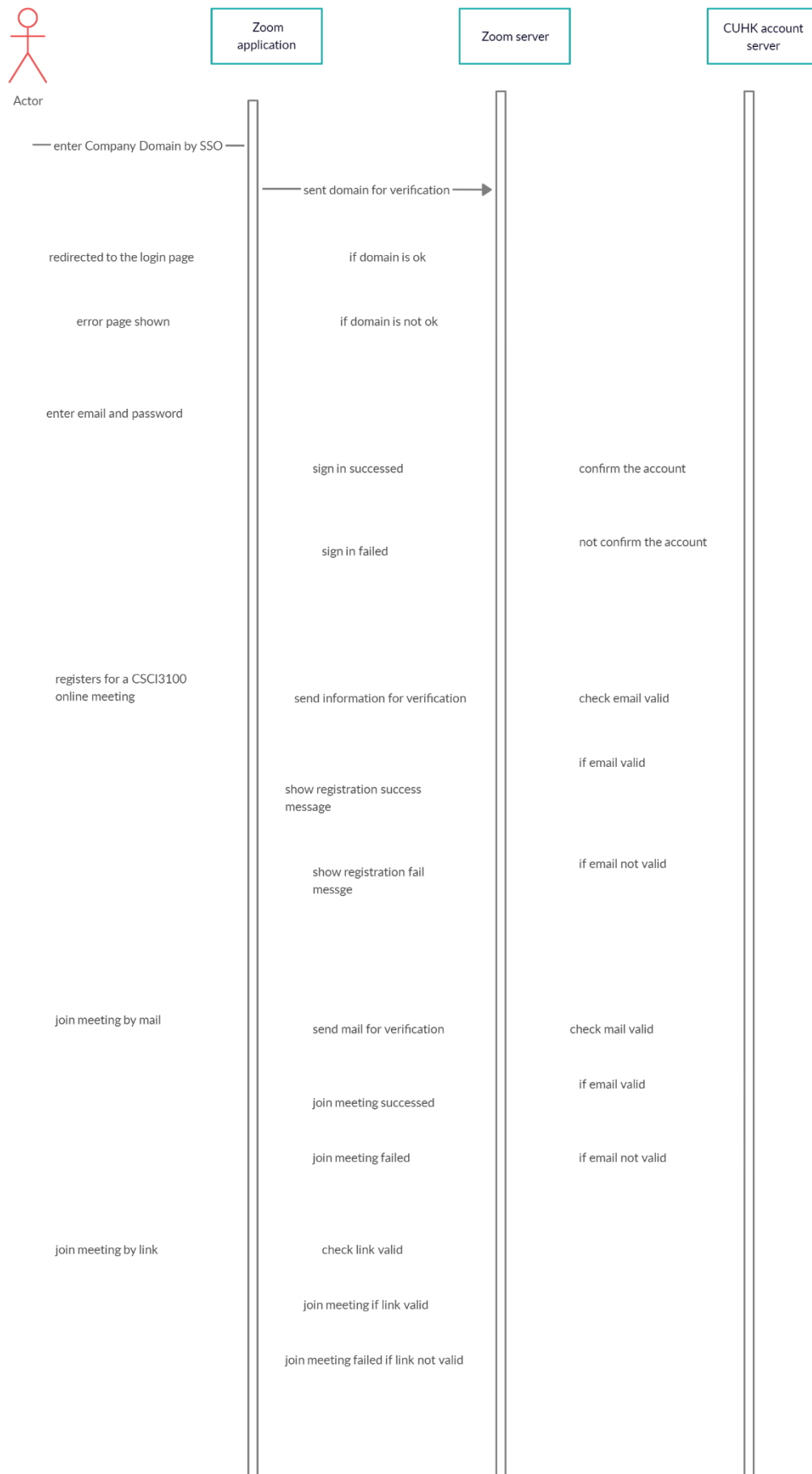
<u>Zoom</u>
- meetinglist : list
- createnewmeeting() : int
- deleteexistingmeeting() : bool

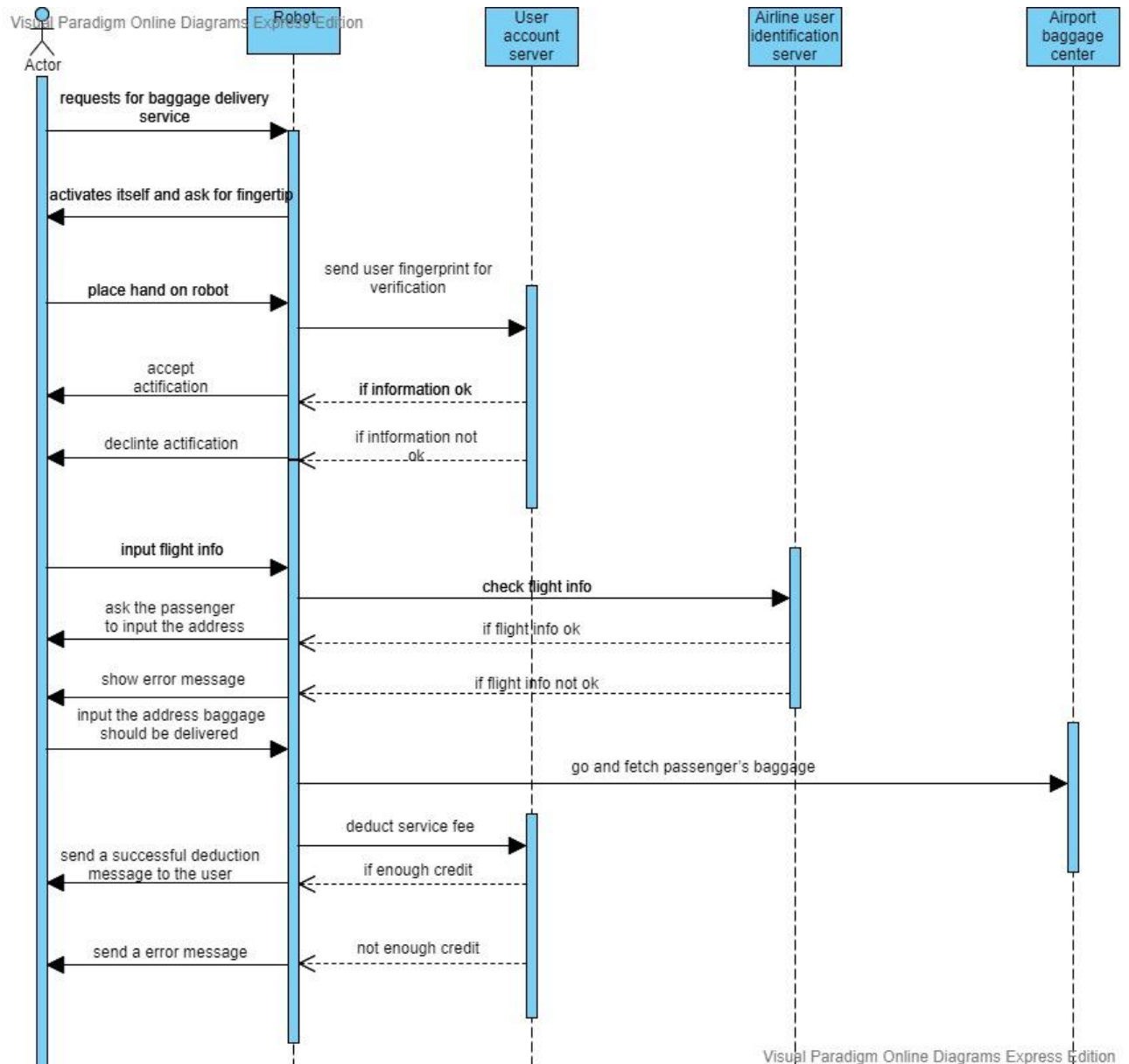
<u>blackborad</u>
- courseslist : list
- addnewcourses() : int
- downloadhomework() : bool
- uploadhomework(): bool

<u>Meeting</u>
- meetingId: int
- invitationlink : string
- Password: string
- getmeetingId: () : int
- getinvitationlink () : string
- joininngmeeting (password) : bool

<u>Course</u>
- courseID: int
- courseName: string
- getcourseID: () : int
- getcourseName () : string

<u>Homework</u>
- courseId: int
- homeworknum : int
- questionlist: list
- gethomeworknum : () : int
- getquestionlist () : list





4 (1)

<i>Private key</i>	<i>Public key</i>	<i>Similarity Score</i>
CAADC	CADB	
CAADC	CADC	
ABCDAB	ACAB	
ABCDAB	ACDAB	
ADBACDDAAB	ADACDDDB	
ADBACDDAAB	ADBCDADAB	

1.3/5 2.4/5 3.4/6 4.5/6 5.7/10 6.6/10

(2)

```
if (m == 0 || n == 0)
```

```
    return 0;
```

```
if (X[m-1] == Y[n-1])
```

```
    return 1 + sim(X, Y, m-1, n-1);
```

```
else
```

```
    return max(lcs(X, Y, m, n-1), sim(X, Y, m-1, n));
```

(3)

```
int max(int a, int b);

int sim( char *X, char *Y, int m, int n )
{
    int L[m + 1][n + 1];

    int i, j;

    for (i = 0; i <= m; i++)
    {
        for (j = 0; j <= n; j++)
        {
            if (i == 0 || j == 0)
                L[i][j] = 0;

            else if (X[i - 1] == Y[j - 1])
                L[i][j] = L[i - 1][j - 1] + 1;

            else
                L[i][j] = max(L[i - 1][j], L[i][j - 1]);
        }
    }

    int main()
    {
        char X[] = "CAADC";
        char Y[] = "CADB";

        int m = strlen(X);
        int n = strlen(Y);

        cout << "Length of sim is "
              << (sim( X, Y, m, n )/ m);

        return 0;
    }
```