ICDE-[Skill Stack]

|  |  |  |
| --- | --- | --- |
| Name: Ajay Kumar Lakshmipura Vijaykumar | Name: Somesh Vemula | Name: Goutham Kumar Prabhakaran |
| Student ID:40168544 | Student ID:40161259 | Student ID:40160167 |
| Program: MEng Electrical and Computer Engineering | Program: MEng Electrical and Computer Engineering | Program: MEng Electrical and Computer Engineering |
| Email: [a\_laks@encs.concordia.ca](about:blank) | Email: [s\_emula@encs.concordia.ca](about:blank) | Email: [g\_prabh@encs.concordia.ca](about:blank) |
| SE Background: Familiar with C, Python and OOPS Concepts | SE Background: Familiar with C, Python and OOPS Concepts | SE Background: Familiar with C, Python and OOPS Concepts |

# Tools Decided

|  |  |
| --- | --- |
| **Description** | **Tool** |
| Code Repository | GitHub |
| IDE | PyCharm |
| Programming Language | Python |
| Group Communication Software | Microsoft Teams |
| Progress Tracking Tool | JIRA |
| Database | MongoDB |

Table 1: Tools

# Problem Statement

*ICDE - [Skill Stack]*

The main aim of the project is to develop a course recommendation system, making it easy for a user to choose a course from the most accessed courses for a particular discipline.

# Architecture Diagram

## High-Level Architecture Diagram:

High-level architecture diagram.

*Fig 1.1 ICDE - [Skill Stack] High-Level Architecture*

Fig.1.1 shows the concrete context of how data flows from multiple users to Database and then to third party applications using ICDE - [Skill Stack].

## System-Level Architecture Diagram:

ICDE - [Skill Stack] System level Architecture

*Fig 1.2 ICDE - [Skill Stack] System level Architecture*

The overview of the system-level architecture is as shown in the above figure (Fig.1.2). This entire application will be hosted on an application server. ICDE data collection client is a part of the ICDE application, which receives data from ICDE users using the ICDE application. The entire collected data will be sent to a third-party application, where the application will further analyze the data for dissemination.

# Key Functionalities

* **Collection of user data**

Accumulating piles of data provided by the user and storing it.

* **Course recommendation based on user's interest**

This course recommendation is based on collecting previously registered users' information from the user data and suggesting the courses to the new users based upon the particular interest and developing skills in the specific discipline.

* **Tracking user actions**

To make the user engaged, tracking the user actions and presenting them to the user.

* **Updating the user profile and storing it in Database**

Storing these user data and preferences in our Database handled by NoSQL Database.

* **Analyzing the actions**

Analyzing the various activities done by the user and storing them in the Database for analysis.

* **Presenting the analyzed data**

Parsing the analyzed data and delivering it statistically with the help of a 3rd party application.

# Use Case Diagram

Diagram

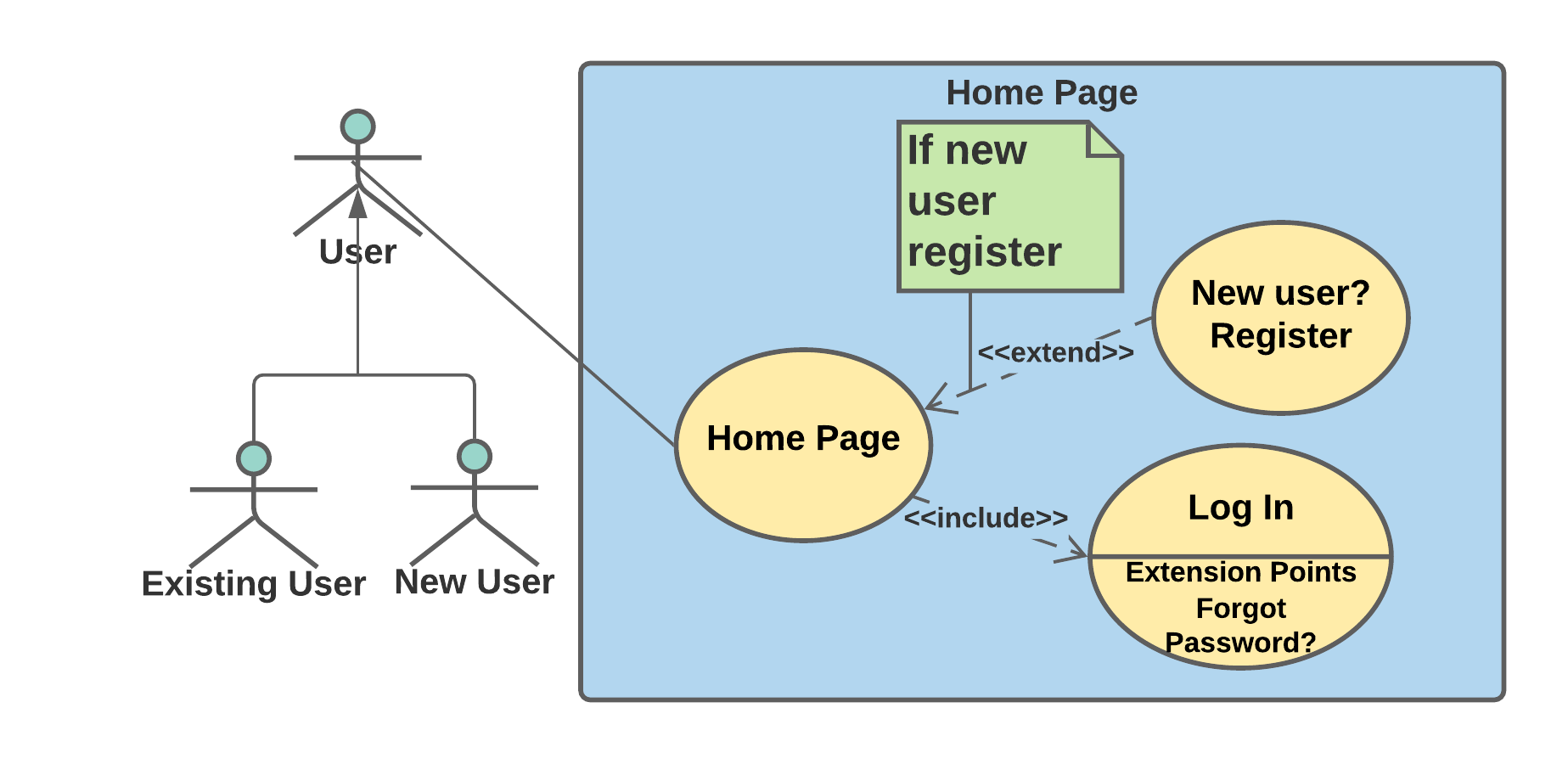
Description automatically generated

*Fig 2. ICDE-Use case Diagram*

# User Stories

*Home Page*

**Use case** - Home Page



*Fig 2.1 Home page use case diagram*

**User Story** - He/ She is the new user and needs to create an account; the fields that are displayed "Login," "Sign Up" and "Exit." If the user selects "Login," the system navigates to the login page, asking for the user credentials. Instead if the user selects "Sign Up," the system asks the user for the user's details for registration. If the user selects "Exit," the system exits the application.

**Requirement Specification** - In the home page, ICDE-Skill Stack System shall have "Sign Up," "Sign In," "Exit" fields. Upon user selection, the ICDE-Skill Stack System shall navigate to the Sign-up page and Sign-In, respectively.

**Sub-Requirements** –

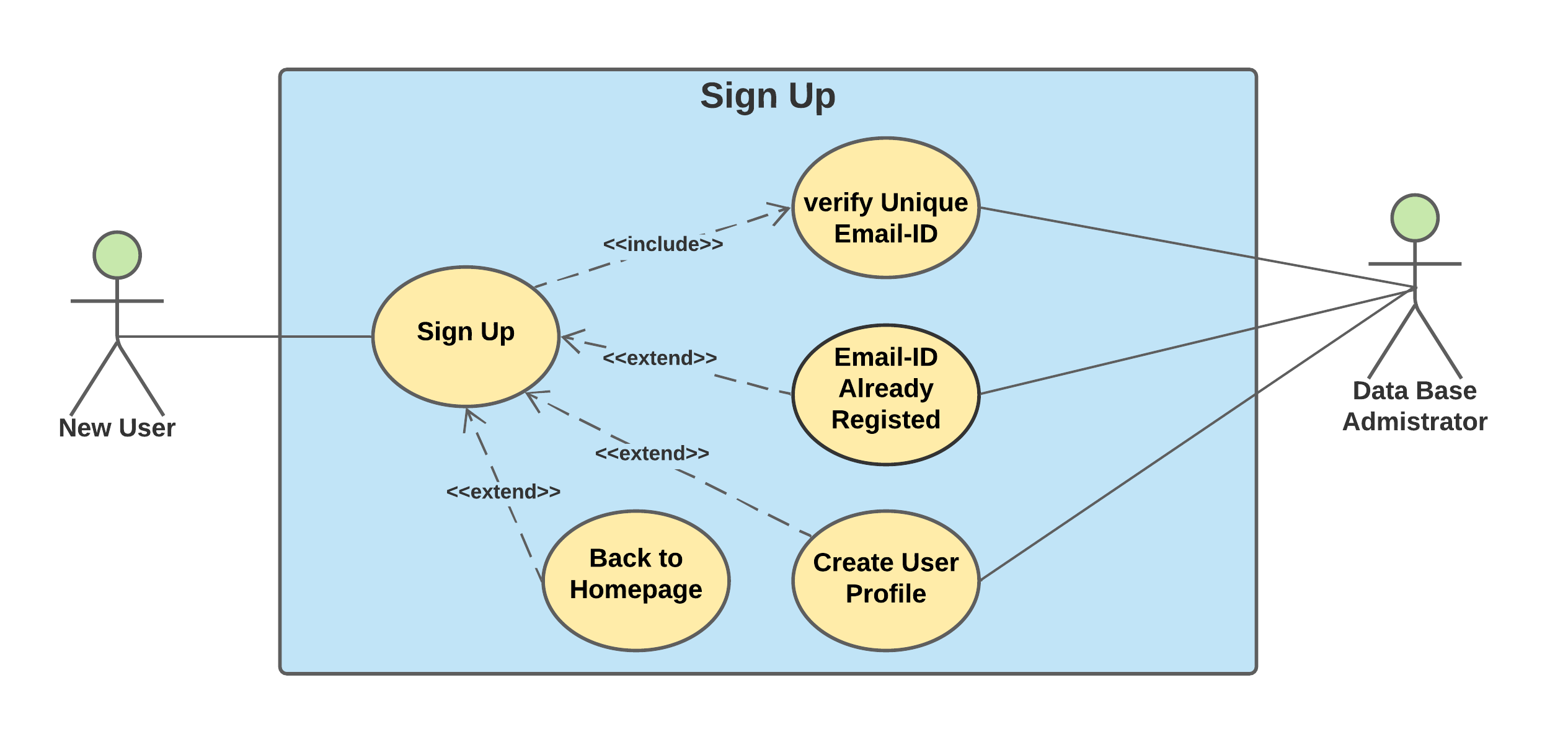
|  |  |
| --- | --- |
| SRS\_**0101** | ICDE-Skill Stack System home page shall have "Sign Up", "Sign In", "Exit" options. |
| SRS\_**0102** | Upon selection, ICDE-Skill Stack System shall navigate to Sign-up page, Sign-In page and Exit the application respectively. |

**Tasks-**

|  |  |
| --- | --- |
| SRT\_**0101** | **Create the home page with "Sign Up" and "Sign-In" options**.  The homepage containing "Sign Up and Sign-In" options is created. |
| SRT\_**0102** | **Navigate to the respective pages**.  On selection in the homepage, the system lands on the respective pages following the choice. |

*Sign Up*

**Use case** - Sign Up



*Fig 2.2 Sign Up page use case diagram*

**User Story** - He/ She is the new user and needs to create an account, the "Sign Up" page displays the fields to collect the details of the user. The user provides all the details such as, "First Name, Last Name, Email, Password, Confirm Password, Security Questions, Mobile Number," and once he clicks "Submit", the account is created. If the user selects "Back to Homepage", he is navigated to the homepage.

**Requirement Specification** - In registration page, ICDE-Skill Stack System shallcollect the user info which is listed below: First Name, Last Name, User Mail ID, Password, Confirm Password, Mobile Number, Discipline. Also, ICDE-Skill Stack System should verify provided User Mail ID is a unique mail id and is not registered before.

**Sub-Requirements** –

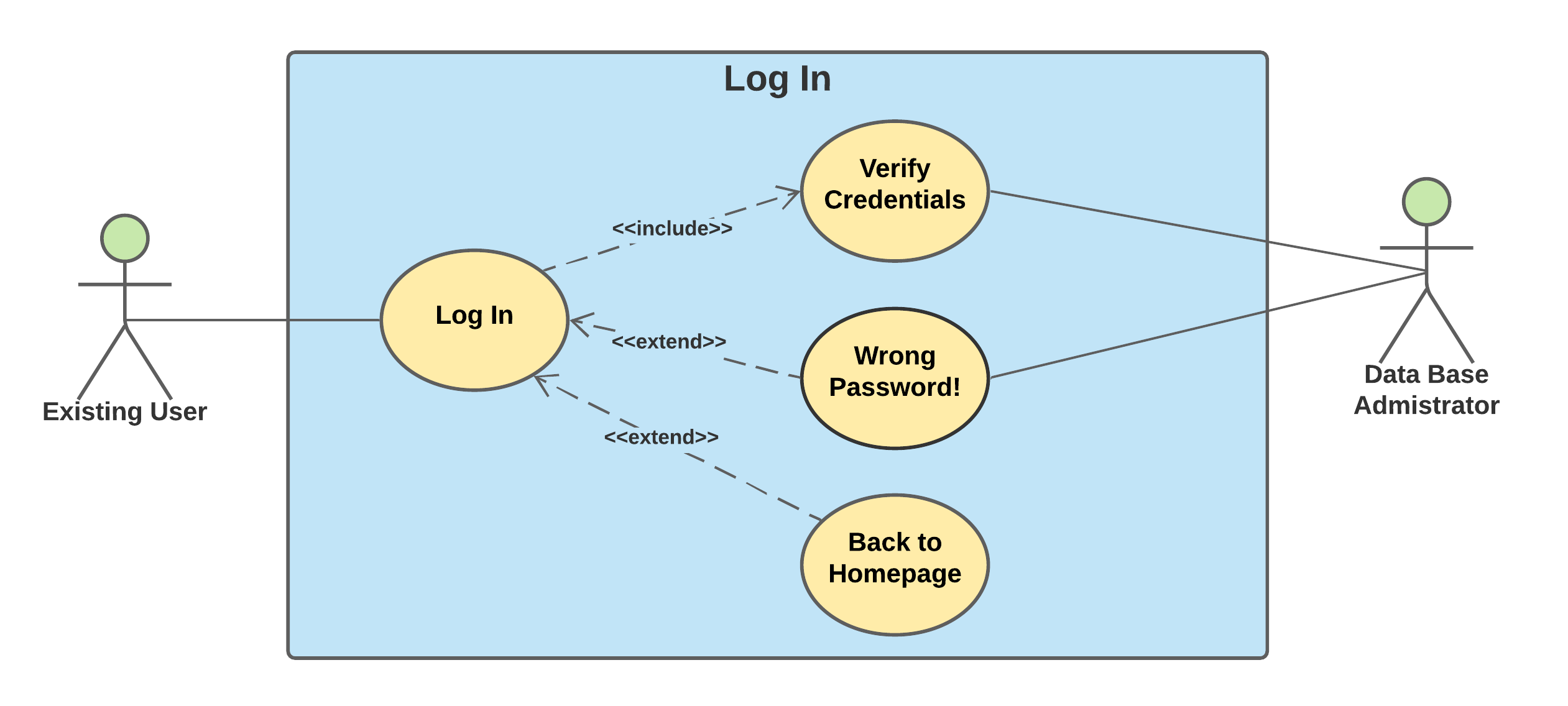
|  |  |
| --- | --- |
| SRS\_**0201** | Registration page shall have fields listed below that are to be collected from a new user: First Name, Last Name, User Mail ID, Password, Confirm Password, Mobile Number, Discipline.  Also, the registration page should have "Back to Homepage" option, which enables the user to return to the home page. |
| SRS\_**0202** | ICDE-Skill Stack System shall verify for unique mail id upon user input. |
| SRS\_**0203** | ICDE-Skill Stack System shall create a user profile and store user data in it. |
| SRS\_**0204** | ICDE-Skill Stack System shall upload user profile to Database. |

**Tasks-**

|  |  |
| --- | --- |
| SRT\_**0201\_01** | **Create registration page**.  The registration page is created to collect the information from the user. |
| SRT\_**0201\_02** | **Collect user info**.  User information for the fields displayed in the registration page will be collected. |
| SRT\_**0201\_03** | **Add "Back to Homepage" option**.  In the registration page, "Back to Homepage" option is created for navigating the user to the application homepage. |
| SRT\_**0202\_01** | **Validate user Email**.  The Email entered is validated for its uniqueness. |
| SRT\_**0202\_02** | **Password hashing**.  The password entered is hashed and stored for the purpose of security |
| SRT\_**0203\_01** | **Create user profile**.  The profile is created upon collecting all the information from the user. |
| SRT\_**0204\_01** | **Store in Database**  The user information collected is stored in the Database. |

*Log In*

**Use case** - Login



*Fig 2.3 Login page use case diagram*

**User Story**-He/ She is a returning user who wish to access the courses, the Login page displays the fields to collect the user credentials. The user provides the user credentials such as, "Email, Password," and once he clicks "Login", the user is logged in. If the user selects "Back to Homepage", he is navigated to the homepage.

**Requirement Specification** - The login page shall have Email and Password fields to fill in by the user. Upon user input ICDE-Skill Stack System verifies the entered credentials and allows the user to log in or display a login error due to the wrong password.

Also, the Login page should have "Back to Homepage" option that enables users to return to the home page.

**Sub-Requirements** –

|  |  |
| --- | --- |
| SRS\_**0301** | The ICDE-Skill Stack System's login page shall have "Email" and "Password" fields to enter by an existing user. Here Username should be user mail id.  Also, the Login page should have a "Back to Homepage" option, which enables the user to return to the home page. |
| SRS\_**0302** | ICDE-Skill Stack System, shall verify the entered credentials and allow user to login or display login. |

**Tasks:**

|  |  |
| --- | --- |
| SRT\_**0301\_01** | **Create a login page with login button**.  The login page which contains the login button is created. |
| SRT\_**0301\_02** | **Collect user credentials**.  The user credentials are collected from the user for logging in. |
| SRT\_**0301\_03** | **Add "Back to Homepage" option**.  "Back to Homepage" is added in order to navigate the user to the homepage from the login page. |
| SRT\_**0302\_01** | **Validate user credentials**.  The user credentials given by the user is validated with the data from Database. |
| SRT\_**0302\_02** | **Navigate to the application main page**.  The user is navigated to the application main page upon validation of the credentials. |

*Reset the password*

**Use case** - Reset the password

Diagram

Description automatically generated

*Fig 2.4 Reset Password page use case diagram*

**User Story** – On account of forgetting the password, there will be a field "Forgot the password?", once selection the user is directed to reset the password where the user will be asked to authenticate his/her email address, and upon successful authentication of the email, a security check will be done, and the user is allowed to reset the password.

**Requirement Specification -** ICDE-Skill Stack System shallcollect the required fields: username, OTP**,** answer for security questionupon successful authentication, he/she can reset the password**.**

**Sub-Requirements** –

|  |  |
| --- | --- |
| SRS\_**0401** | ICDE-Skill Stack System shall have forgot password option on login screen. |
| SRS\_**0402** | Forgot password page shall have a security check to authenticate the user email address. |
| SRS\_**0403** | Upon successful authentication, ICDE-Skill Stack System shall take new password from user. |
| SRS\_**0404** | ICDE-Skill Stack System shall update a user profile with an updated password and store it in Database. |

**Tasks -**

|  |  |
| --- | --- |
| SRT\_**0401\_01** | **Add "Forgot Password" option to login page**.  "Forgot Password" is added in the login page in case if the user forgets the password. / The login page must have Forgot Password option in case if the user forgets the password. |
| SRT\_**0402\_01** | **Create a Reset Password page**.  Reset Password page must have security questions fields and New Password fields to collect from the user. |
| SRT\_**0403\_01** | **Add OTP generation option.**  Upon giving the email address, system has to generate a OTP and send that OTP to user's email address. |
| SRT\_**0403\_02** | **Create user authorisation page.**  System shall accept the new password from the user and updates it with the old one. |
| SRT\_**0403\_03** | **Password hashing in forgot password page**.  The password entered is hashed and stored for the purpose of security. |
| SRT\_**0404\_01** | **Update the user Profile.**  The user profile is then updated with the new set of information given by the user. |

*Course/Workshop selection*

**Use case -** Course/Workshop selection

*Diagram

Description automatically generated*

*Fig 2.5 Course/Workshop selection use case diagram*

**User Story**- User who wish to select workshop/ course, once he lands on the main page there will be three fields – "General workshops", "Technical Courses" & "Non-Technical Courses". If the user selects "General workshops", he'll then be the given various options to choose according to his convenience. If the user selects for "Technical", "Non-Technical", Courses, the user is given various options under different titles to choose from.

**Requirement Specification** - Once after successful login, ICDE-Skill Stack System shall have a main page displaying with General Workshop, Technical course and Non-Technical course sections so that user can select a course from the list of courses inside these course sections conveniently.

**Sub-Requirements** –

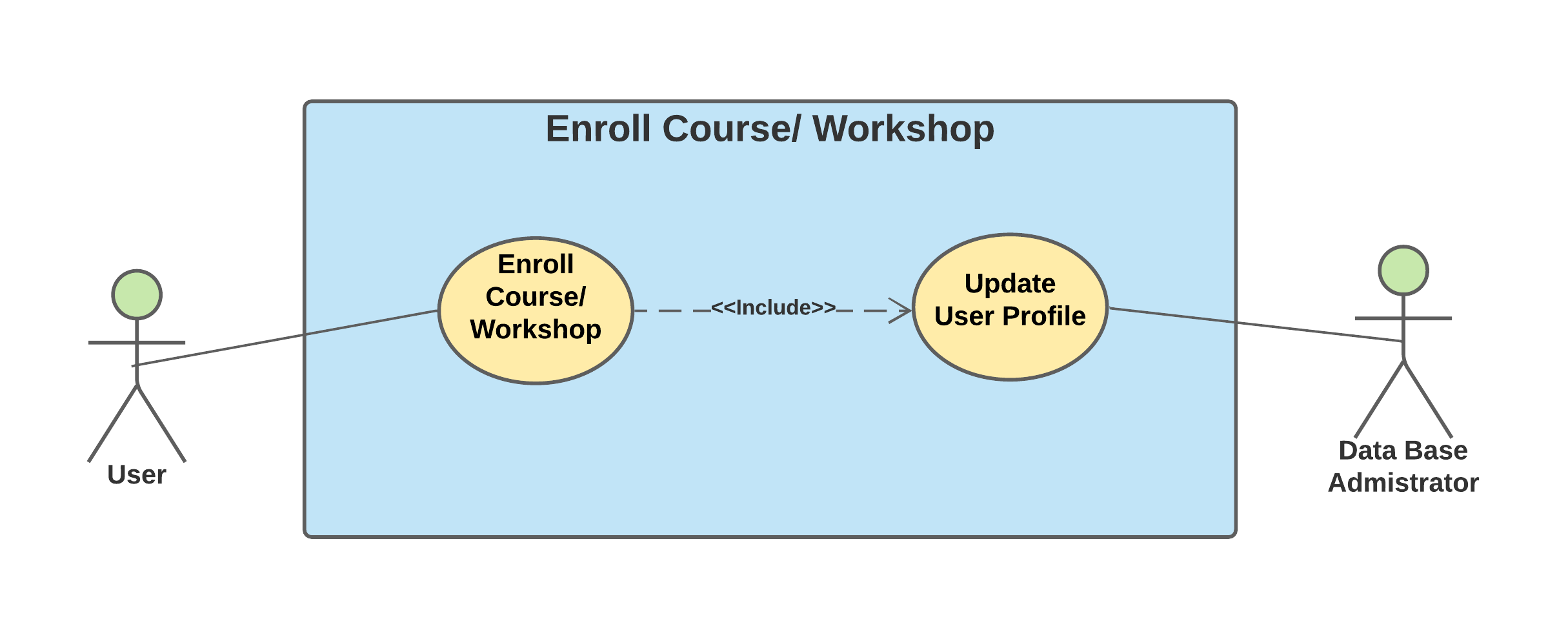
|  |  |
| --- | --- |
| SRS\_**0501** | ICDE-Skill Stack System shall have main page displaying General Workshop, Technical course and Non-Technical course sections. |
| SRS\_**0502** | ICDE-Skill Stack System Shall allow user to select courses under General Workshop, Technical course and Non-Technical course sections. |
| SRS\_**0503** | ICDE-Skill Stack System shall have a profile section in the application main page. Profile section shall display last accessed date and time, user's name, logout option, and my usage stastics button. |

**Tasks-**

|  |  |
| --- | --- |
| SRT\_**0501\_01** | **Create application main page.**  Create application main page with General Workshop, Technical course and Non-Technical course sections display. |
| SRT\_**0502\_01** | **Create course section page.**  Create a page that displays list of related courses under that section.   1. Create General Workshop course section page. 2. Create Technical course section page. 3. Create Non-Technical course section page. |
| SRT\_**0502\_02** | **Navigate to enroll/drop page.**  The user is navigated to a page where he/she can enroll/drop the courses upon the selection of the course. |
| SRT**\_0503\_01** | **Create profile section in the application main page.**  Create profile section in the application main page with "logout" button and "my usage stastics" button. |
| SRT**\_0503\_02** | **Display last accessed date and time in the profile section.**  Query the last accessed date and time from Database and display it in the profile section. |
| SRT**\_0503\_03** | **Display user statistical graph upon clicking "my usage stastics" button.**  Query user's usage stastics from the data base and display it to the user in the form of a graph. |
| SRT**\_0503\_04** | **Navigate to login page upon clicking "Logout" button.**  Navigate to the login page when the user selected "Logout" option. |

*Enrollment*

**Use case -**  Enrollment



*Fig 2.6 Enrollment use case diagram*

**User Story** – After the user decides on the course/workshop, he/she will be able to access the course only after enrolling. Thus, there will be a field "Enroll". If the user selects "Enroll" he/she is enrolled. The course enrolled is updated to user profile.

**Requirement Specification** - As user selects a course from the list of courses, the application shall have an enroll option for a selected course and then update the enrolled course onto user profile.

**Sub-Requirements** –

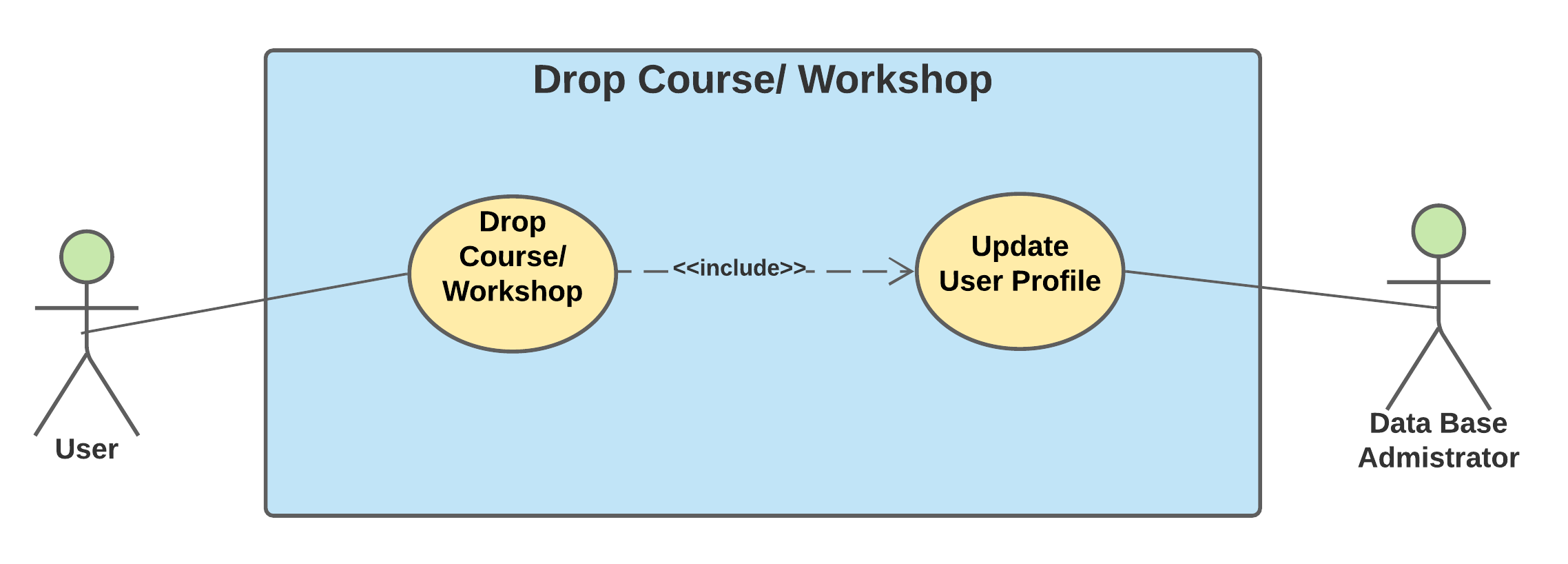
|  |  |
| --- | --- |
| SRS\_**0601** | ICDE-Skill Stack System shall take the response from user to enroll for a particular course. |
| SRS\_**0602** | ICDE-Skill Stack System shall update the enrolled course onto user profile. |

**Tasks -**

|  |  |
| --- | --- |
| SRT\_**0601\_01** | **Create Enroll/Drop page**.  The enroll/drop page is created for the purpose of modifying the selection of the course by the user. |
| SRT\_**0601\_02** | **Create enroll option**.  Enroll in to course upon clicking on enroll option if the user has not enrolled for the selected course before. |
| SRT\_**0601\_03** | **Check for enrollment**  Display pop-up message "already enrolled to the course" upon clicking on enroll option if the user has enrolled for the selected course before. |
| SRT\_**0602\_01** | **Update the user profile in Database**.  The user profile is updated in the Database in accordance to the selection made by the user. |

*Drop a course/workshop*

**Use case -** Drop a course/workshop

**

*Fig 2.7 Drop a course/workshop use case diagram*

**User Story** – Upon enrollment, the user is also given the option of dropping the course at any time. So, if he/she decides to drop the course, there will be a field in every course, "Drop the course". On selecting it, the user is asked for the reason for dropping it, after reviewing, the user is successfully dropped from the particular course.

**Requirement Specification** - For every course/workshop enrolled by the user, the application shall have drop the course option.

**Sub-Requirements** --

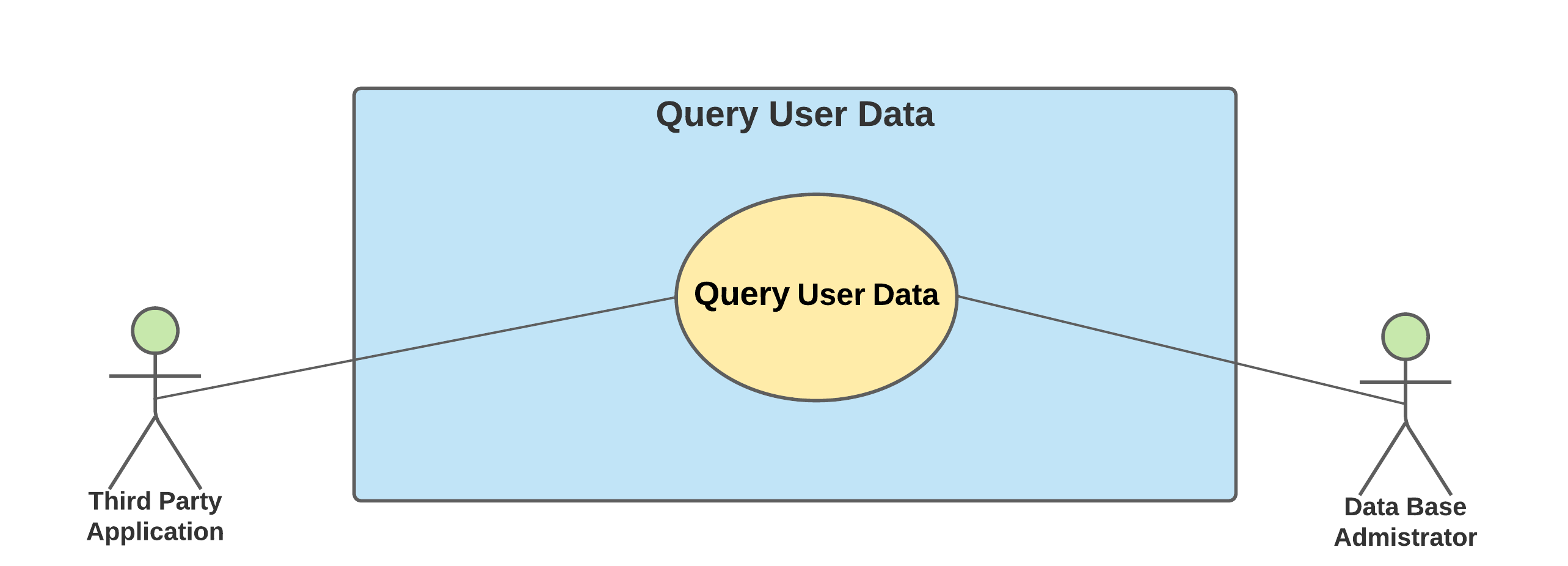
|  |  |
| --- | --- |
| SRS\_**0701** | ICDE-Skill Stack System shall take the response from user to drop a particular course. |
| SRS\_**0702** | ICDE-Skill Stack System shall update the user profile upon dropping a course. |

**Tasks -**

|  |  |
| --- | --- |
| SRT\_**0701\_01** | **Create Drop option**  Drop the course upon clicking on Drop option if user had enrolled for selected course before. |
| SRT\_**0701\_02** | **Check for enrollment**  Display pop up message "Not enrolled to the course" upon clicking on Drop option if user had not enrolled for selected course before. |
| SRT\_**0702\_01** | **Update the information in Database**  The user profile is updated in the Database in accordance to the selection made by the user. |

*Query User Data*

**Use case -** Query user data

****

*Fig 2.8 Query user data use case diagram*

**User Story** - To present the statistical report on user actions, the ICDE system queries the data of various users registered in the same discipline and sends it to the 3rd party application.

**Requirement Specification** - As per the user's selection, the application **shall** query the enrolled workshops and courses list of users of the same discipline from the Database and passing it to the 3rd party application.

**Sub-Requirements –**

|  |  |
| --- | --- |
| SRS\_**0801** | ICDE-Skill Stack System shall query the list of enrolled courses from database sorting user with the same discipline. |
| SRS\_**0802** | ICDE-Skill Stack System shall pass the queried data to the 3rd party application |

**Tasks -**

|  |  |
| --- | --- |
| SRT\_**0801\_01** | **Sorting the course**  Sort registered courses among users of the same discipline. |
| SRT\_**0802\_01** | **Communicate it with 3rd party**  Pass sorted data to third party applications. |

*Recommend courses*

**Use case** – Recommend courses to user

Diagram

Description automatically generated

*Fig 2.9 Recommend courses to user use case diagram*

**User Data** - He/she is in search of a trending course in his/her discipline. In view of achieving it, the administrator who queries the data, analyzes it and presents the data according to certain categories making it easy for the user to select. Those fields include, "Most accessed courses/workshops".

**Requirement Specification** - Third-party Application shallcollect the queried data and analyze it to draw a pattern and return the results.

**Sub-Requirements –**

|  |  |
| --- | --- |
| SRS\_**0901** | Third party application shall need to analyze the queried data. |
| SRS\_**0902** | ICDE-Skill Stack System shall present the analyzed data to the user. |

**Tasks-**

|  |  |
| --- | --- |
| SRT\_**0901\_01** | **Analyze and store the data**  Collect the analyzed data from third party application and store it in Database. |
| SRT\_**0902\_01** | **Recommend courses to users.**  The course is recommended to the user according to the analyzed data. |

# Product Backlog

|  |  |
| --- | --- |
| **Sl. No** | **Task ID** |
|  | SRT\_0101\_01 |
|  | SRT\_0102\_01 |
|  | SRT\_0201\_01 |
|  | SRT\_0201\_02 |
|  | SRT\_0201\_03 |
|  | SRT\_0202\_01 |
|  | SRT\_0202\_02 |
|  | SRT\_0203\_01 |
|  | SRT\_0204\_01 |
|  | SRT\_0301\_01 |
|  | SRT\_0301\_02 |
|  | SRT\_0301\_03 |
|  | SRT\_0302\_01 |
|  | SRT\_0302\_02 |
|  | SRT\_0401\_01 |
|  | SRT\_0402\_01 |
|  | SRT\_0403\_01 |
|  | SRT\_0403\_02 |
|  | SRT\_0403\_03 |
|  | SRT\_0404\_01 |
|  | SRT\_0501\_01 |
|  | SRT\_0502\_01 |
|  | SRT\_0502\_02 |
|  | SRT\_0503\_01 |
|  | SRT\_0503\_02 |
|  | SRT\_0503\_03 |
|  | SRT\_0503\_04 |
|  | SRT\_0601\_01 |
|  | SRT\_0601\_02 |
|  | SRT\_0601\_03 |
|  | SRT\_0602\_01 |
|  | SRT\_0701\_01 |
|  | SRT\_0701\_02 |
|  | SRT\_0702\_01 |
|  | SRT\_0801\_01 |
|  | SRT\_0802\_01 |
|  | SRT\_0901\_01 |
|  | SRT\_0902\_01 |

*Table 2: Product Backlog*

# Team setup & Sprint cycle

Overall tasks : 38

Timeline : 7 Weeks

Duration of 1-Sprint Cycle : 1 Week

Minimum No of Tasks/Week : 5

The sprint cycle is followed according to the above-given details. Each sprint cycle, an average of six tasks are chosen to complete. The sprint cycle works/functions/operates five days a week, including work for 2 hours per day to the developer, an additional 30 minutes dedicated for scrum meetings. The changes required to the software design and documentation will be taken up after discussion in scrum meetings, and then both the product owner and scrum master include work of 1 hour per day to update these changes and document them.

*Week 1 - 15/02/2021 to 19/02/2021*

**Team member roles:**

Product Owner : Goutham Kumar Prabhakaran

Scrum-Master : Ajay Kumar Lakshmipura Vijaykumar

Developer : Somesh Vemula

**Tasks Taken Up:**

Below mentioned tasks are taken up for week 1 in prioritized order.

|  |  |
| --- | --- |
| **Sl. No** | **Task ID** |
| 1 | SRT\_0101\_01 |
| 2 | SRT\_0201\_01 |
| 3 | SRT\_0201\_03 |
| 4 | SRT\_0202\_02 |
| 5 | SRT\_0301\_01 |
| 6 | SRT\_0301\_02 |

**Tasks Completed:**

All the tasks mentioned below are completed in prioritized order.

|  |  |
| --- | --- |
| **Sl. No** | **Task ID** |
| 1 | SRT\_0101\_01 |
| 2 | SRT\_0201\_01 |
| 3 | SRT\_0201\_03 |
| 4 | SRT\_0202\_02 |
| 5 | SRT\_0301\_01 |
| 6 | SRT\_0301\_02 |

*Week 2 - 22/02/2021 to 26/02/2021*

**Team member roles:**

Product Owner : Somesh Vemula

Scrum-Master : Goutham Kumar Prabhakaran

Developer : Ajay Kumar Lakshmipura Vijaykumar

**Tasks Taken Up:**

Below mentioned tasks are taken up for week 2 in prioritized order.

|  |  |
| --- | --- |
| **Sl. No** | **Task ID** |
| 1 | SRT\_0202\_02 |
| 2 | SRT\_0302\_01 |
| 3 | SRT\_0301\_03 |
| 4 | SRT\_0302\_02 |
| 5 | SRT\_0401\_01 |
| 6 | SRT\_0402\_01 |
| 7 | SRT\_0403\_01 |

**Tasks Completed:**

All the tasks mentioned below are completed in prioritized order.

|  |  |
| --- | --- |
| **Sl. No** | **Task ID** |
| 1 | SRT\_0202\_02 |
| 2 | SRT\_0302\_01 |
| 3 | SRT\_0301\_03 |
| 4 | SRT\_0302\_02 |
| 5 | SRT\_0401\_01 |
| 6 | SRT\_0402\_01 |
| 7 | SRT\_0403\_01 |

*Week 3 - 01/03/2021 to 05/03/2021*

**Team member roles:**

Product Owner : Ajay Kumar Lakshmipura Vijaykumar

Scrum-Master : Somesh Vemula

Developer : Goutham Kumar Prabhakaran

**Tasks Taken Up:**

Below mentioned tasks are taken up for week 3 in prioritized order.

|  |  |
| --- | --- |
| **Sl. No** | **Task ID** |
| 1 | SRT\_0403\_03 |
| 2 | SRT\_0501\_01 |
| 3 | SRT\_0502\_02 |
| 4 | SRT\_0502\_01 |
| 5 | SRT\_0601\_01 |
| 6 | SRT\_0601\_02 |

**Tasks Completed:**

All the tasks mentioned below are completed in prioritized order.

|  |  |
| --- | --- |
| **Sl. No** | **Task ID** |
| 1 | SRT\_0403\_03 |
| 2 | SRT\_0501\_01 |
| 3 | SRT\_0502\_02 |
| 4 | SRT\_0502\_01 |
| 5 | SRT\_0601\_01 |
| 6 | SRT\_0601\_02 |

# Modeling and Design of Tasks

**Completed task:**

SRT\_**0301\_02** - Collect user credentials.

The user credentials are collected from the user for logging in.

## Class Diagram:

## Class Diagram for task SRT_0301_02

*Fig 9.1 Class Diagram for task SRT\_****0301\_02***

## Sequence Diagram:

Sequence Diagram for task SRT_0301_02

*Fig 9.2 Sequence Diagram for task SRT****\_0301\_02***

## Activity Diagram:

Activity Diagram for task SRT_0301_02

*Fig 9.3 Activity Diagram for task SRT****\_0301\_02***

## Statement Diagram:

Statement Diagram for task SRT_0301_02

*Fig 9.4 Statement Diagram for task SRT****\_0301\_02***

**New task:**

SRT**\_0203\_01** - Create a user profile.

The profile is created upon collecting all the information from the user.

## Class Diagram:

*Diagram

Description automatically generated*

*Fig 9.5 Class Diagram for task SRT\_****0203\_01***

## Sequence Diagram:

*Sequence Diagram for task SRT_0203_01*

*Fig 9.6 Sequence Diagram for task SRT****\_0203\_01***

## Activity Diagram:

*Activity Diagram for task SRT_0203_01*

*Fig 9.7 Activity Diagram for task SRT****\_0203\_01***

## Statement Diagram:

Statement Diagram for task SRT_0203_01

*Fig 9.8 Statement Diagram for task SRT\_****0203\_01***

# Evaluation

Overall tasks : 34

Completed tasks : 19

Remaining tasks : 15

Remaining Timeline : 4 Weeks

Duration of 1-Sprint Cycle : 1 Week

Minimum No of Tasks/Week : 4

As the number of tasks that need to be completed in remaining 4 weeks of time duration is 15, the minimum number of tasks per week is reduced to 4 tasks/week. Since, the tasks that will be taken up in future might take bit longer duration for the development and documentation of changes to the design. Therefore, the scrum working hours were kept the same as before.

# Architectural Designs

The system's software architecture is the basic structure of the system that contains different software elements and relationships among them. Software architecture must be designed to fulfill specific requirements and constraints of the application which is being developed.

In our case, ICDE- [Skill Stack] is a desktop application, so in the database perspective, the application is distributed across multiple systems, and a constraint is that certain functionalities must reside in the desktop application. To develop an architecture model for any system, we need to address so many issues like system requirements, constraints involved in the design, technical issues related to the system, user base of the system, type of system developed, etc. To address our system's software processes, we are dividing the architecture design constraints into two categories as follows.

* Technical constraints.
* Business constraints.

**Technical constraints:** As the name suggests, these are the constraints of the system's technical aspect. Some of the technical aspects we have considered while designing our system architecture are discussed below.

Satisfying functional requirements is one of the significant aspects of the architectural model. One of the functional requirements of the system of our system is data collection from the ICDE user. As shown in *Fig 1.2 ICDE - [Skill Stack] System level Architecture,* we have accommodated the ICDE data collection client for collecting and passing data to the central app server. We can directly connect the ICDE data collection client with the Database, but it will lead to tight coupling and more dependencies.

Another functional requirement is to analyze the collected data. As shown in fig-1.2,we have accommodated the ICDE data access client for managing and passing data from the central app server to third-party tools. We can directly connect the ICDE data access client with the Database, but again it will lead to tight coupling.

Another functional requirement is to disseminate the analyzed data to the ICDE user. We have decomposed the data dissemination service for disseminating the analyzed data from the Database to ICDE users.

In any application,systemstate changes are expected. Suppose the system architecture is tightly coupled together. In that case, a state change in one system forces all the systems connected to that particular system to change their states, and this process propagates throughout the entire architecture. So, by minimizing dependencies, changes are localized and thereby creating a loosely coupled architecture. As you can see in fig-1.2, ICDE data collection client and ICDE data access client are not directly connected with the Database. Our design tried to form a loosely coupled architecture by reducing dependencies.

Our system's top three core services are data collection service, data accessing service through third-party applications, and the data dissemination service. These are the services where almost every user will access these services. Most of the time, these services have to interact with each other for passing and collecting data from the ICDE user. So, we have decomposed the most commonly accessed services into a component, as shown in fig-1.2.

**Business constraints:** As the name suggests, these are the constraints of the system's non-technical aspect. Some of the business aspects we have considered while designing our system architecture are discussed below.

One of the significant constraints that will affect the entire project is the budget.Since our project is at the university level and a non-commercial application, we have selected the free tire MongoDB database, which is providing a NoSQL database with limited storage of 500MB for free.

# Layered Architecture

The gradual construction of structures is supported by this layered approach. Any of the services offered by a layer will be made accessible to users as it develops. In addition, the architecture is adaptable and compact. This layered architecture has User interface layer, Configuration layer, Application processing layer, and Database layer. Each layer will user the functionality of the below layer.



*Fig 9.9 Layered Architecture*

In Fig **9.9**, is modeled using a layered approach with the user interface at the top and the system database at the bottom. The Database selected in our application is No SQL database (MongoDB). In the Database layer, all the application and user data will be maintained. The application processing layer has all the application-specific functionalities, and it uses the functionality of the database layer to perform its operations. Registering user, event registration functions are used to subscribe a user to the application and maintain the user profile. Data access and Data query will be performed to collect the data and perform data computation on 3rd party application. The 3rd party application will do data analysis and disseminate the analyzed data as course recommendation. The Configuration layer will identify users and allows them to subscribe and use the functionalities of the application. This layer will maintain user management functions, user authentication, data collection, and data validation. The user interface layer allows users to interact with the application.

# MVC Architecture

MVC (model–view–controller) is a software design pattern that divides related program logic into three interconnected elements and is widely used for designing user interfaces. This is done to distinguish internal knowledge representations from how information is presented to and embraced by users.



*Fig 9.10 MVC Architecture*

In Fig **9.10,** the Controller is responsible for carrying out the functions based upon the user selection. In ICDE-Skill Stack, if the user selects any option that is presented to him, say "Login" option, the user is then asked for the login credentials. The view is responsible for the display of the selected frames upon getting input from the Controller. The model contains the overall data the application requires.

The MVC pattern is incorporated in ICDE-Skill Stack, with the Controller performing the functions according to the user input. The view block contains the user interfaces that are designed with this system, it includes,

* The Sign in/Sign up page
* The application main page
* The course selection page
* Enrollment/Drop page

The Controller also queries/updates the user information in the model block(Database). The relationship between the Controller and view is generally associated with two parameters, the Display and User Action. Upon receiving the user action, the Controller sends a signal to the view to display the required page (UI).

# Design decisions-Pros and Cons

**Architectural pattern design decision**

1. **Problem:** What Architectural patterns or styles might be used for designing the archicture of ICDE[SkillStack]?

**Motivation:** Presenting and using the knowledge of software systems that are practically proven and commonly used.

**Cause:** Understandability of the system-level architecture.

**Context:** The layered design of the ICDE[SkillStack] system is shown in fig-9.9.

**Potential solutions:**

* **Layered architecture**

**Description:** Organizes the structure into layers, with each layer containing associated features. The lowest level layers reflect key services that are likely to be used in the system since each layer offers services to the layer above it.

**Pros:** Allows whole layers to be replaced as long as the interface is preserved. To improve the system's dependability, redundant facilities (such as authentication) can be given at each layer.

**Cons:** In reality, maintaining a clean layer separation can be challenging, and a high-level layer can have to communicate with lower-level layers directly rather than via the layer immediately below it.

* **Repository architecture**

**Description:** All system data is stored in a central repository that is open to all of the system's components. Components do not interact directly with one another; instead, they communicate via the repository.

**Pros:** Components may be self-contained; they don't need to be aware of other components' presence. All components may be affected by changes made by one component. Since all data is in one location, it can be handled consistently (for example, backups can be performed at the same time).

**Cons:** Since the repository is a single point of failure, any issues there have an impact on the whole system. There could be inefficiencies in the repository's organization of all communications.

**Decision:** The pattern used in our design is the layered pattern. Allows whole layers to be replaced as long as the interface is preserved.

**Component decomposition design decision**

**Problem:** There are many modules in the ICDE[SkillStack] application; some modules should be decomposed to form a compact architecture.

**Motivation:** The need for compact architecture design.

**Cause:** ICDE[SkillStack] users should use the application effectively.

**Context:** The design of the ICDE[SkillStack] system is shown in fig-1.2.

**Potential solutions:**

* **No decomposition**

**Description:** ICDE[SkillStack] system adopts various systems and modules to fulfill the design. System state changes are required in any program. Assume that the system architecture is inextricably linked. In that case, a state transition in one system causes all the systems related to it to change their states as well, and the process propagates across the architecture.

**Pros:** Minimum level of decomposition will lead to an easily maintainable system and lesser maintenance costs.

**Cons:** Tightly coupled architecture, changes are globalized across the entire architecture.

* **Decomposition**

**Description:** If some of the structural components are decomposed, Changes are localized as a result of eliminating dependencies, resulting in a loosely coupled architecture. The ICDE data collection client and ICDE data access client are not directly linked to the Database, as seen in fig-1.2. By eliminating dependencies, we attempted to create a loosely coupled architecture.

**Pros:** Loosely coupled architecture, Changes are localized for the respective components.

**Cons:** Maximum-level of decomposition will lead to long-term maintainability problems.

**Decision:** Our system's top three core services are data collection service, data accessing service through third-party applications, and the data dissemination service. These are the services to which almost every user would have access. Most of the time, these services must communicate with one another in order to pass and collect data from ICDE users. So, we have decomposed the most commonly accessed services into a component, as shown in fig-1.2.

# Software Metrics and Granularity of Components

The components' granularity is at the level of class methods. The task name column will have all the tasks derived from all the user stories. The LOC represents "Lines of Code" written for implementing that corresponding task. The "Number of units" indicates the number of class methods that implement the respective task.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Sl.No | Task Name | LOC | Component Granularity Level | Number of Units |
| 1 | Create home page with “Sign Up” and “Sign-In” options. | 16 | Python Class (ICDEUser), Method- homepage | 1 |
| 2 | Navigate to the respective pages. | 98 | Python Class (ICDEUser), Method- homepage | 1 |
| 3 | Create registration page | 62 | Python Class (ICDEUser), Method - registration page | 1 |
| 4 | Collect user info | 143 | Python Class (ICDEUser), Method - registration page, Python Class (Functionalities), Method-signup\_data\_validation | 2 |
| 5 | Add “Back to Homepage” option | 6 | Python Class (ICDE User), Method-registration page, Login page | 2 |
| 6 | Validate user Email. | 8 | Python Class (Functionalities), Method - login\_data\_validation | 1 |
| 7 | Password hashing. | 3 | Python Class (Data\_Verification), Method -PwHashing | 1 |
| 8 | Create user profile. | 28 | Python Class (IcdeAppCore), Method-createProfile | 1 |
| 9 | Store in database | 7 | Python Class (IcdeDataBase), Method- create\_profile\_db | 1 |
| 10 | Create a login page with login button. | 6 | Python Class (ICDEUser), Method-loginpage | 1 |
| 11 | Collect user credentials. | 12 | Python Class (ICDEUser), Method - loginpage | 1 |
| 12 | Add “Back to Homepage” option. | 6 | Python Class (ICDE User), Method-registration page, Login page | 2 |
| 13 | Validate user credentials | 25 | Python Class (Functionalities), Method - login\_data\_validation | 1 |
| 14 | Navigate to application main page | 55 | Python Class (ICDEUser), Method - AppMainPage | 1 |
| 15 | Add "Forgot Password" option to login page. | 5 | Python Class (ICDEUser), Method - loginpage | 1 |
| 16 | Create a Reset Password page. | 28 | Python Class (ICDEUser), Method-passwordResetFrame | 1 |
| 17 | Validate answers to security questions. | 6 | Python Class (Data Verification), Method-VerifySeqQn | 1 |
| 18 | Add OTP generation option for Password reset | 22 | Python Class (Functionalities), Method-generateotp, authorisefn | 2 |
| 19 | Create user authorisation page for password reset | 22 | Python Class (ICDEUser), Method-authenticatepage | 1 |
| 20 | Collect new password from user | 13 | Python Class (ICDEUser), Method-passwordResetFrame | 1 |
| 21 | Password hashing in forgot password page. | 3 | Python Class (Data\_Verification), Method -PwHashing | 1 |
| 22 | Update the user Profile | 4 | Python Class (IcdeDataBase), Method- update\_pw\_db | 1 |
| 23 | Create application main page. | 57 | Python Class (ICDEUser), Method - AppMainPage | 1 |
| 24 | Create course section page. | 340 | Python Class (ICDEUser), Method-Courseselection | 1 |
| 25 | Navigate to enroll/drop page | 5 | Python Class (ICDEUser), Method- EnrollmentPage | 1 |
| 26 | Create Enroll/Drop page. | 15 | Python Class (ICDEUser), Method- EnrollmentPage | 1 |
| 27 | Create enroll option | 4 | Python Class (ICDEUser), Method- EnrollmentPage | 1 |
| 28 | Check for enrolment | 5 | Python Class (IcdeAppCore), Method-check\_enrollement | 1 |
| 29 | Update the user profile in Database | 12 | Python Class (IcdeDataBase) | 1 |
| 30 | Create Drop option | 4 | Python Class (ICDEUser), Method- EnrollmentPage | 1 |
| 31 | Check for enrolment | 5 | Python Class (IcdeAppCore), Method-check\_enrollement | 1 |
| 32 | Update the information in Database | 13 | Python Class (IcdeDataBase), Method-collect\_courses\_data | 1 |
| 33 | Communicate it with 3rd party | 21 | Python Class (Graphs), Method-plot\_course\_ranks | 1 |
| 34 | Analyse and store the data | 26 | Python Class (IcdeDataBase), Method-plot\_usage\_stats | 1 |
| 35 | Recommend courses to users | 33 | Python Class (IcdeDataBase), Python Class (Graphs), Method-plot\_course\_ranks,collect\_courses\_data | 2 |

*Table 3: Component Granularity Table*