Software Design Document

CS 345-346

Problem 2 : Classnote Taking Application

Group 18

Name	Roll Number
Param Aryan Singh	180101055
Parth Dhananjay Bakare	180101056
Pooja Gajendra Bhagat	180101057
Tejas Prashant Khairnar	180101081

Contents

1 Introduction

1.1 Purpose

This software design document provides a low-level description of the **Classnote Taking Application**, providing insight into the structure and design of each component. It also shows how the functions described in the SRS will be implemented, and helps the reader visualise the platform's operation.

1.2 Intended Audience and Reading Suggestions

This software design document is a part of a project under the **CS345** and **CS346** courses - Software Engineering. The intended audience is our course instructor - **Dr. Samit Bhattacharya**, and teaching assistants tasked with evaluating this project. It is meant to be used by skilled software professionals for understanding the **Classnote Taking Application**.

1.3 Product Scope

Sincerely taking notes during lectures helps a student center and better comprehend the principle ideas being taught by the professor. Note-making is a core value diligently followed by an ideal student, and helps him/her recollect the learnings and reproduce and apply them when needed to.

This application aims to provide a one-stop destination to a student that satisfies his/her needs to take notes digitally. This system eases the process of not only making notes, but also efficiently organising them in a manner best suited to being used by the student as and when needed. When used effectively, the application acts as an extension of the student's mind, recording his/her thought processes and saving them in the form of electronic notes for future use. The student can make notes in various forms provided by the application.

1.4 References

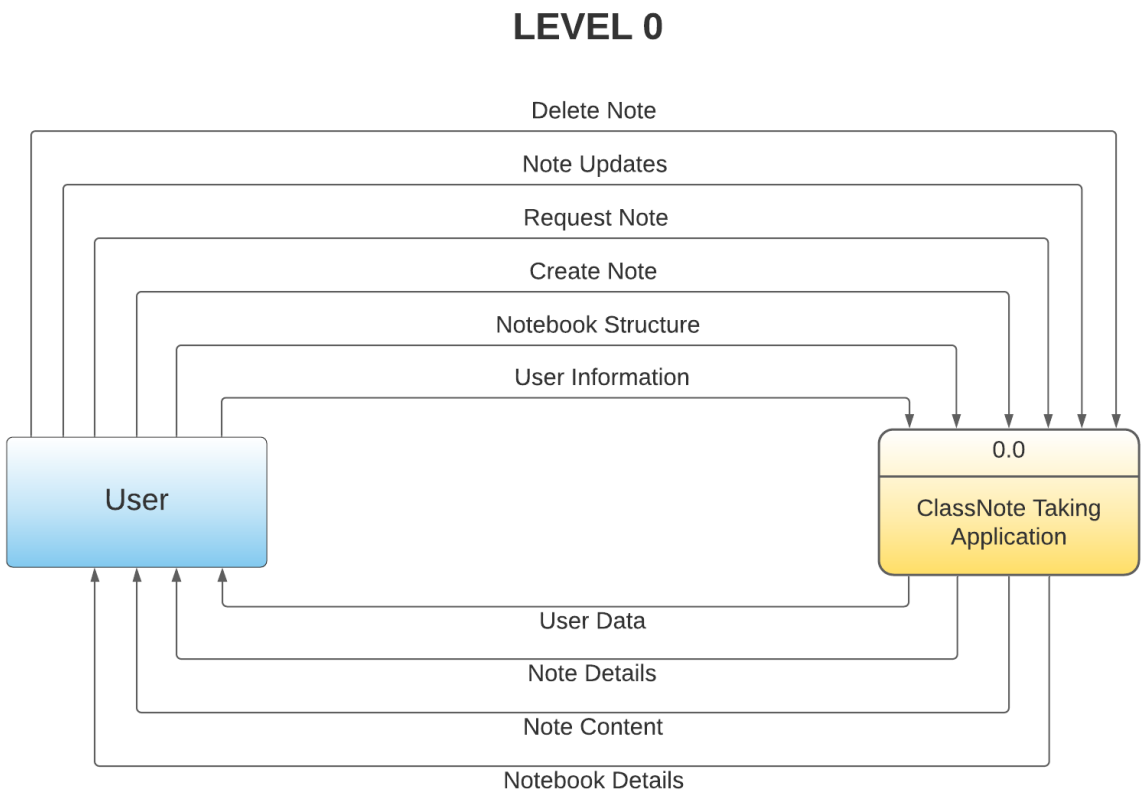
1. IEEE Software Design Document (SWDD) Template
2. CS345 and CS356 course slides by Dr. Samit Bhattacharya. (Winter Semester 2021)

2 System Overview

Classnote Taking Application aims to provide a one-stop destination to a student that satisfies his/her needs to take notes digitally. It provides a platform to the user to efficiently create, organize, manage and visualise his/her notes from multiple devices.

The user is required to create an account on the application using his/her email and then he/she can log in to the application from any device and access his/her notes. The notes can be placed in separate notebooks so that the user can neatly organize the notes and also keep related notes together. The user can create multiple notes, view an existing note and update the content of a previously created note. The notes can include text content and tables as well as graphic content such as images and hand-drawings.

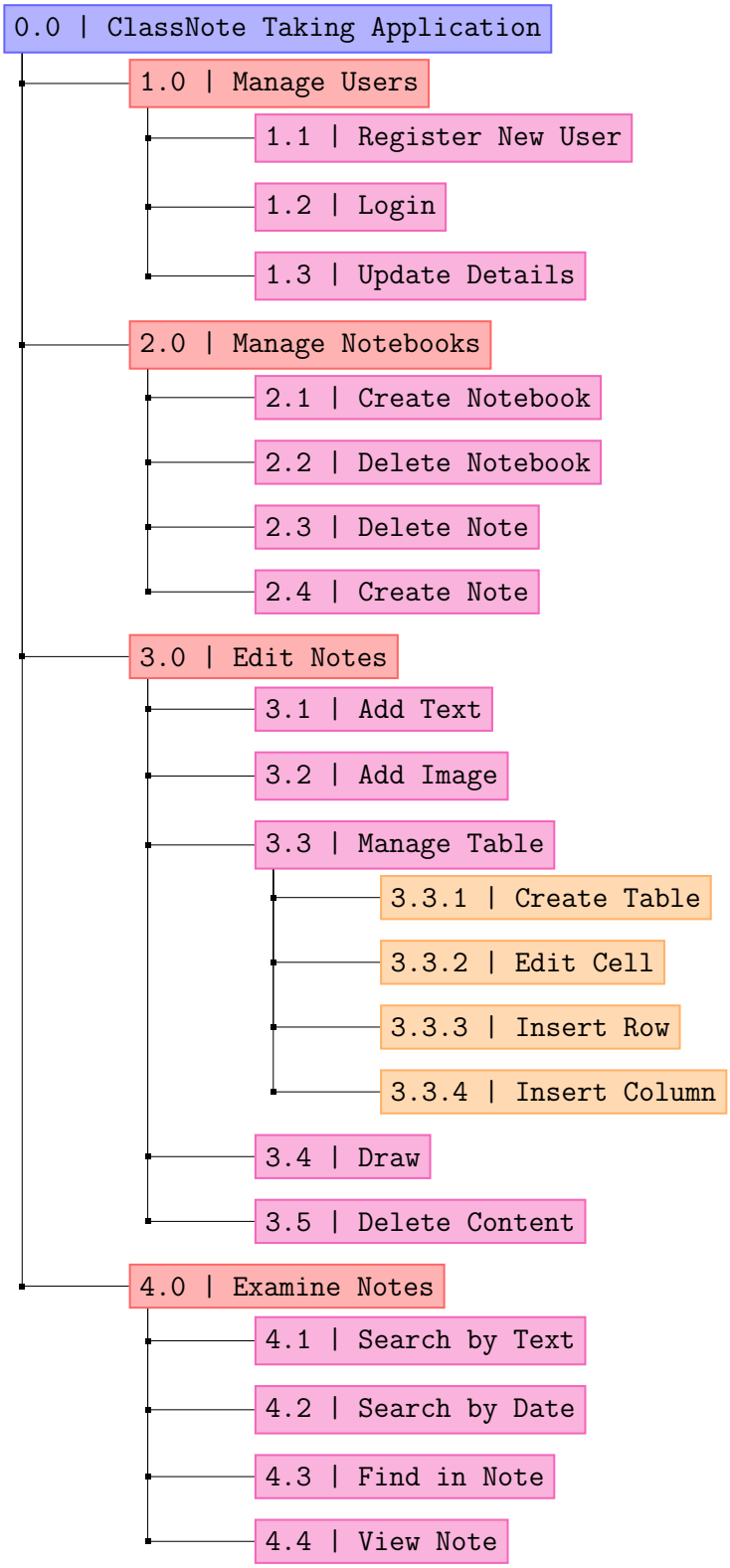
The user can interact with the system as shown in the below
Data Flow Diagram (DFD) Level 0 diagram:



3 System Architecture

3.1 Architectural Design

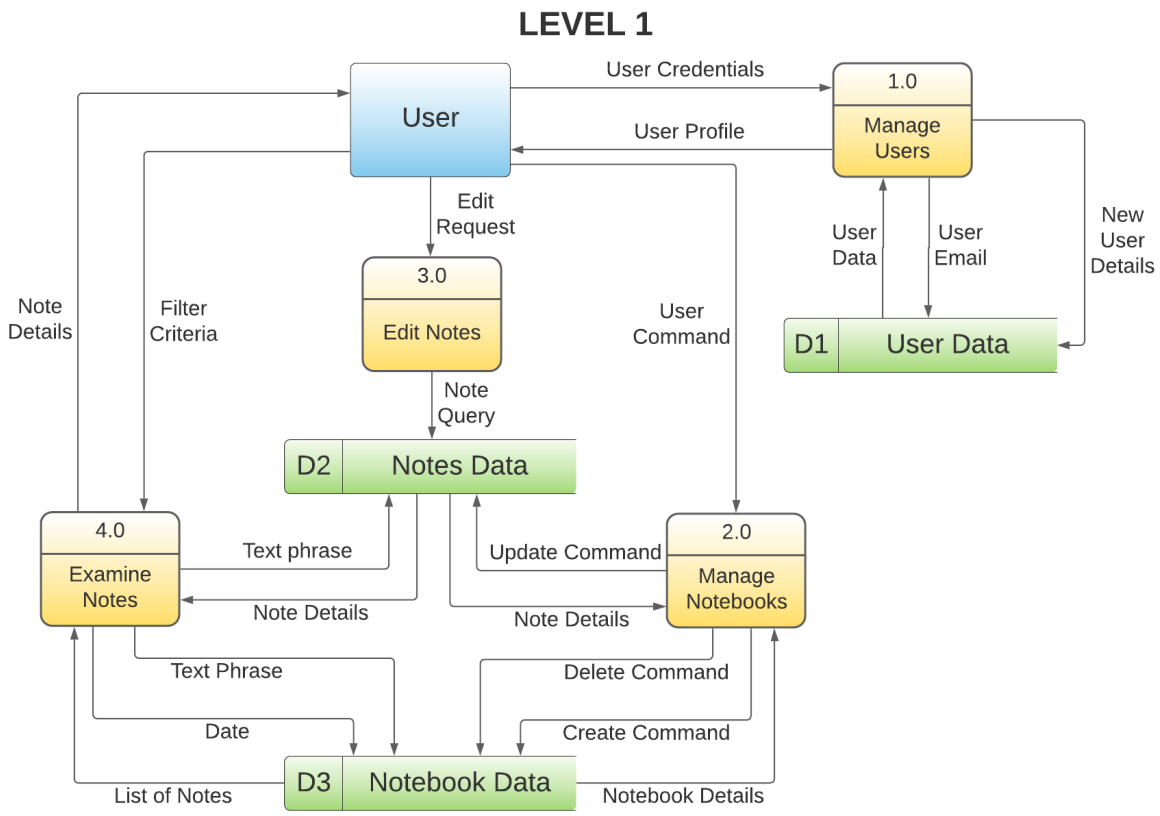
The figure below shows the decomposition of our **Classnote Taking Application**.



There are 4 main modules:

1. **Manage Users:** This module deals with creation of user accounts and user authentication to access data.
2. **Manage Notebook:** This module deals with structurally organizing the user data in the form of Notebooks and Notes.
3. **Edit Notes:** This module deals with modifying the note content, which includes both text and media content.
4. **Examine Notes:** This module deals with parsing the notes and giving the user relevant content as per his/her requirement which involves filtering notes - searching by text or searching by date. It also provides the functionality to find keywords inside a note.

The interaction between the modules and the data stores are as shown in the below **Data Flow Diagram (DFD) Level 1** diagram:



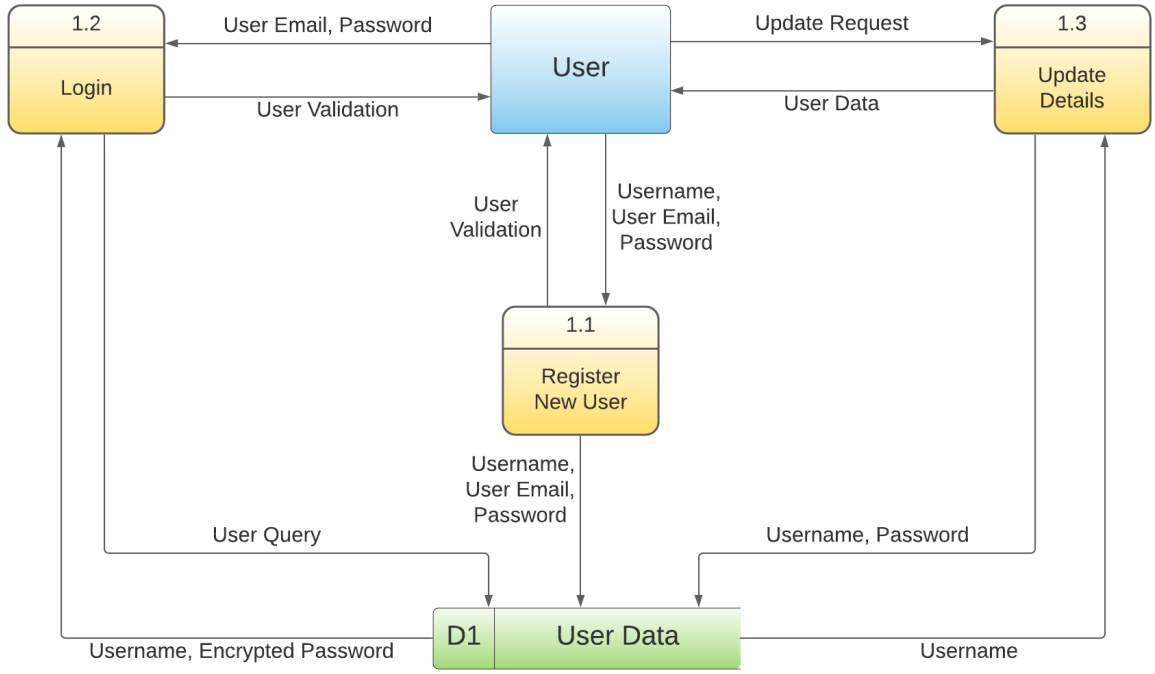
3.2 Decomposition Description (Data Flow Diagram)

3.2.1 Module 1 - Manage Users

There are 3 main processes in this module :

1. **Register New User:** This process deals with registration of a new user wherein username, user email and password to be set are taken as input from the user. A new entry is created in the User Data (Data Store) and the user receives a validation message from the process.
2. **Login:** This process deals with the authentication of an existing user wherein the user email and password is taken as input and verified with the User Data (Data Store). An appropriate validation message is sent to the user.
3. **Update Details:** This process deals with the updation of username or password of a user wherein the new username or the new password is taken as an update request by the process and is subsequently updated in the User Data (Data Store).

LEVEL 2 , Module 1 : Manage Users

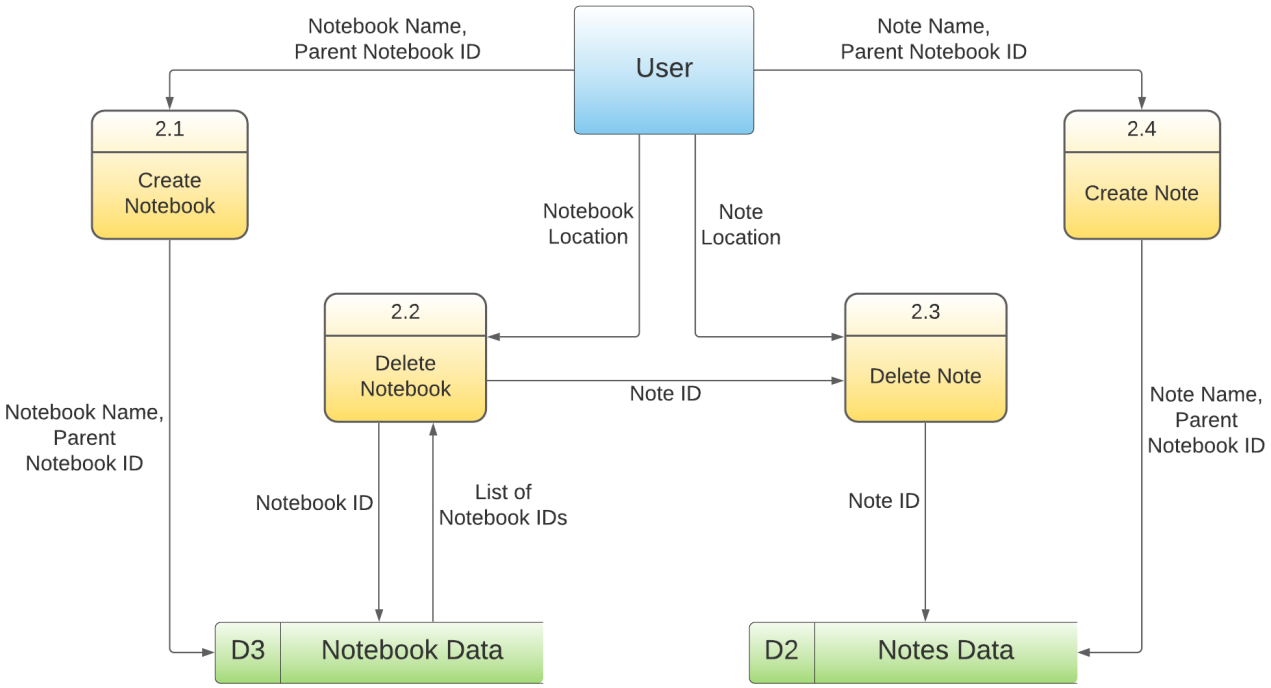


3.2.2 Module 2 - Manage Notebooks

There are 4 main processes in this Module :

1. **Create Notebook:** This process deals with creation of a new notebook. The user gives the location and name for the new notebook and a new entry is created in Notebook Data (Data Store) and it's ID is returned to the process.
2. **Delete Notebook:** This process deals with deletion of a notebook. The user gives the location and name of the notebook to be deleted. The process searches for the corresponding entry in Notebook Data as well as the entries of notes and notebooks contained in it, and all those entries are deleted recursively.
3. **Create Note:** This process deals with creation of a new note. The user gives the location and name for the new note and a new entry for an empty notebook is created in Notes Data (Data Store) and it's ID is returned to the process.
4. **Delete Note:** This process deals with deletion of a note. The user gives the location and name of the note to be deleted. The process searches for the corresponding entry in Notes Data and that entry is deleted.

LEVEL 2 , Module 2 : Manage Notebooks

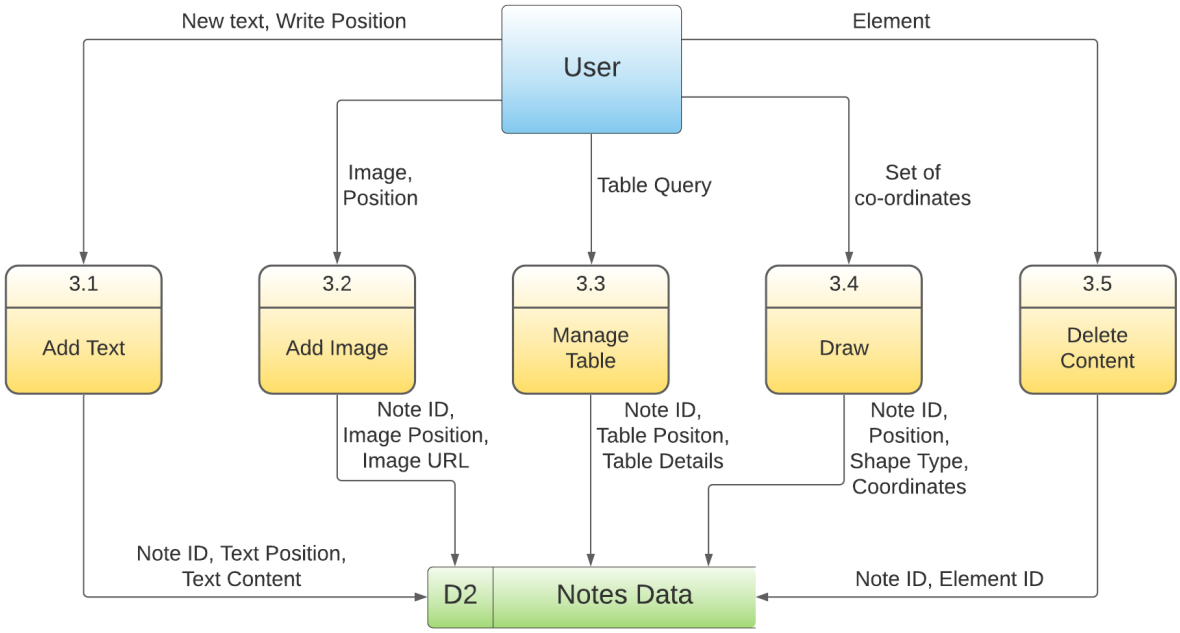


3.2.3 Module 3 - Edit Notes

There are 5 main processes in this Module :

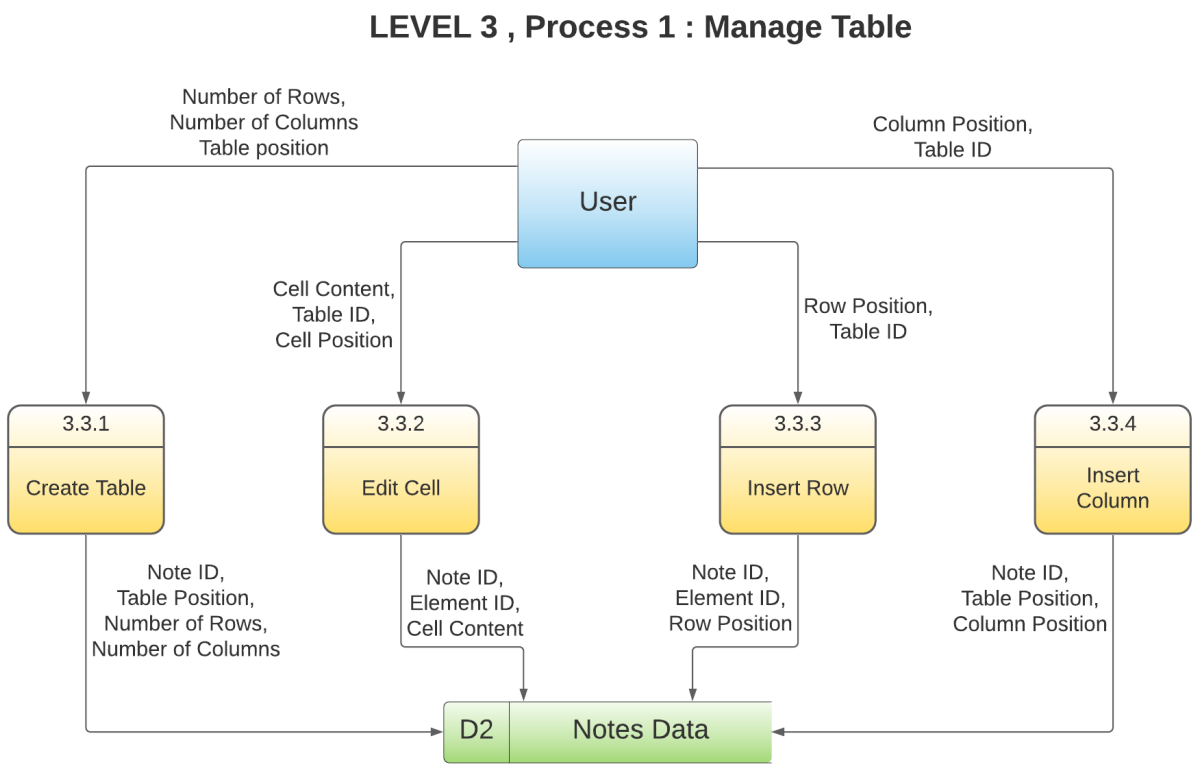
1. **Add Text:** This process deals with taking the write position and the new text as an input from the user and updates the same in the Notes Data (Data Store).
2. **Add Image:** This process deals with taking the image and the position of the image to be inserted as an input from the user and updates the same in the Notes Data (Data Store).
3. **Manage Table:** This process deals with taking a table query, be it creation of table, adding a row/column or editing a cell of a column and updates the same in the Notes Data (Data Store).
4. **Draw:** This process deals with drawing of shapes or annotations and takes a set of co-ordinates as an input from the user and the process updates the same in the Notes Data (Data Store).
5. **Delete Content:** This process deals with deletion of contents from the note. It takes the element to be deleted as an input from the user and then updates the same in Notes Data (Data Store).

LEVEL 2 , Module 3 : Edit Notes



3.2.3.1 Manage Table

1. **Create Table:** This process deals with the creation of a new table. The user passes the table position, number of rows and number of columns as input to create the table.
2. **Edit Cell:** This process deals with the editing of a cell's contents. The row number and column number of the cell along with the new text are passed as input and the corresponding cell is edited.
3. **Insert Row:** This process deals with inserting a new row, at the desired position in the table.
4. **Insert Column:** This process deals with inserting a new column, at the desired position in the table.

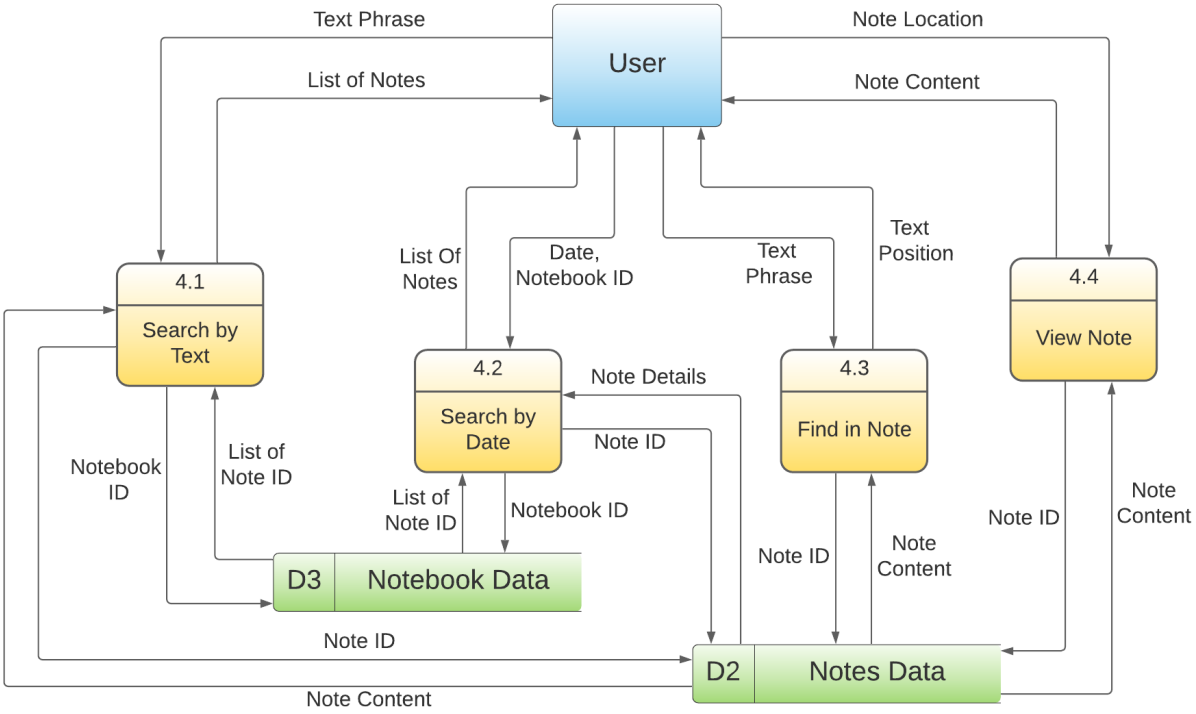


3.2.4 Module 4 - Examine Notes

There are 4 main processes in this Module :

1. **Search by Text:** This process gives the list of notes present inside the notebook specified by the user which contain the text given as input by the user. The process fetches the content of all the notes present inside the notebook and returns the list of notes which match the criteria.
2. **Search by Date:** This process gives the list of notes present inside the notebook specified by the user which are created on the date given as input by the user. The process fetches all the notes present inside the notebook and returns the list of notes which match the criteria.
3. **Find in Note:** This process finds if the given text phrase is present inside the note. The user gives the text phrase to the process and the process fetches the note content from Notes Data (Data Store). The process then goes through the whole text content and returns the position at which the text matches with the given text phrase.
4. **View Note:** This process deals with displaying the notes to user. The user gives the location of the note and the the process fetches its content from Notes Data (Data Store), and displays the content to the user.

LEVEL 2 , Module 4 : Examine Notes



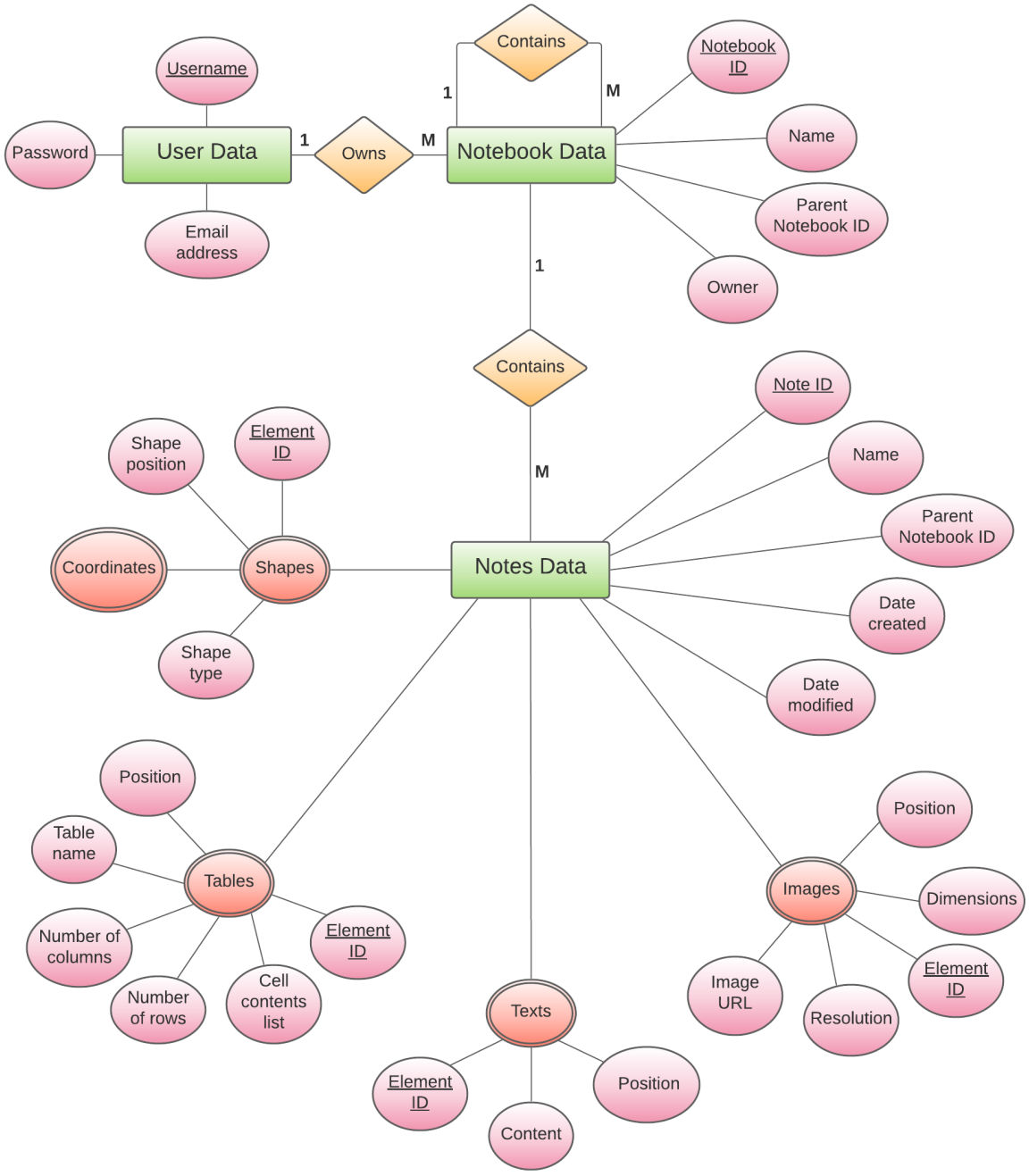
3.3 Design Rationale

The proposed architectural design in section 3.1 efficiently divides all the expected functionalities amongst different modules. It takes **Shneiderman's eight golden rules** into account to provide good usability. The created modules also balance **Cohesion and Coupling** principles. **Thorough justifications of these claims are in the attached documents**.

4 Data Design

4.1 Data Description

The following **entity-relationship diagram** graphically illustrates how data would be stored and used in the application. The rectangles represent the actual data stores (entity sets), the ellipses attached to each data store represent its attributes that characterise it, and the diamonds describe the relationship between data stores. This entity-relationship diagram follows the **standard ER diagram conventions**.



4.2 Data Dictionary

4.2.1 User Data

It stores the credentials of every user who uses this application and assists in managing access. It has the following attributes-

- **Username:** Account name set by the user. Username will be uniquely assigned so that no two users have the same username.
- **Password:** Ensures protection so that a user can safely access his/her notes.
- **Email address:** Every account will be linked with one email address. This would ease accessing the account.

4.2.2 Notebook Data

It stores the details of all the notebooks which is the basic structural component of the application and helps neatly organize notes. A user can own multiple notebooks, and every notebook can contain multiple notebooks and notes inside it. A notebook has the following attributes-

- **Notebook ID:** Uniquely identifies the notebook.
- **Name:** Name assigned to the notebook by the user, similar to a directory name.
- **Parent Notebook ID:** ID of the notebook inside which this notebook resides - foreign key.
- **Owner:** Username of the user who owns the notebook - foreign key.

4.2.3 Notes Data

A note is the actual file which contains the notes written by the user. Every note is stored inside a parent notebook. It has the following attributes-

- **Note ID:** Uniquely identifies the note.
- **Name:** Name of the note, set by the user.
- **Parent Notebook ID:** ID of the notebook inside which this note resides - foreign key.
- **Date Created:** Date when the note was created.
- **Date Modified:** Date when the note was last modified.
- **Texts:** Textual data in the form of strings. It is a multi-valued field, since every note can contain multiple texts. Every text element is further characterised by its unique element ID (inside the note), position (inside the note) and the actual text the element contains.
- **Images:** Images added in the note by the user. It is a multi-valued field, since every note can contain multiple images. Every image is further characterised by its unique element ID (inside the note), position (inside the note), dimensions, resolution and a URL which redirects to the actual image.

- **Tables:** Tables created by the user. It is a multi-valued field, since every note can contain multiple tables. Every table is further characterised by its unique element ID (inside the note), position (inside the note), table name, number of rows and columns and an ordered list of every cell's content.
- **Shapes:** Shapes drawn by the user. It is a multi-valued field, since every note can contain multiple shapes. Every shape is further characterised by its unique element ID (inside the note), position (inside the note), shape type (rectangle, circle, free-hand etc) and the set of coordinates lying on the perimeter of the shape.