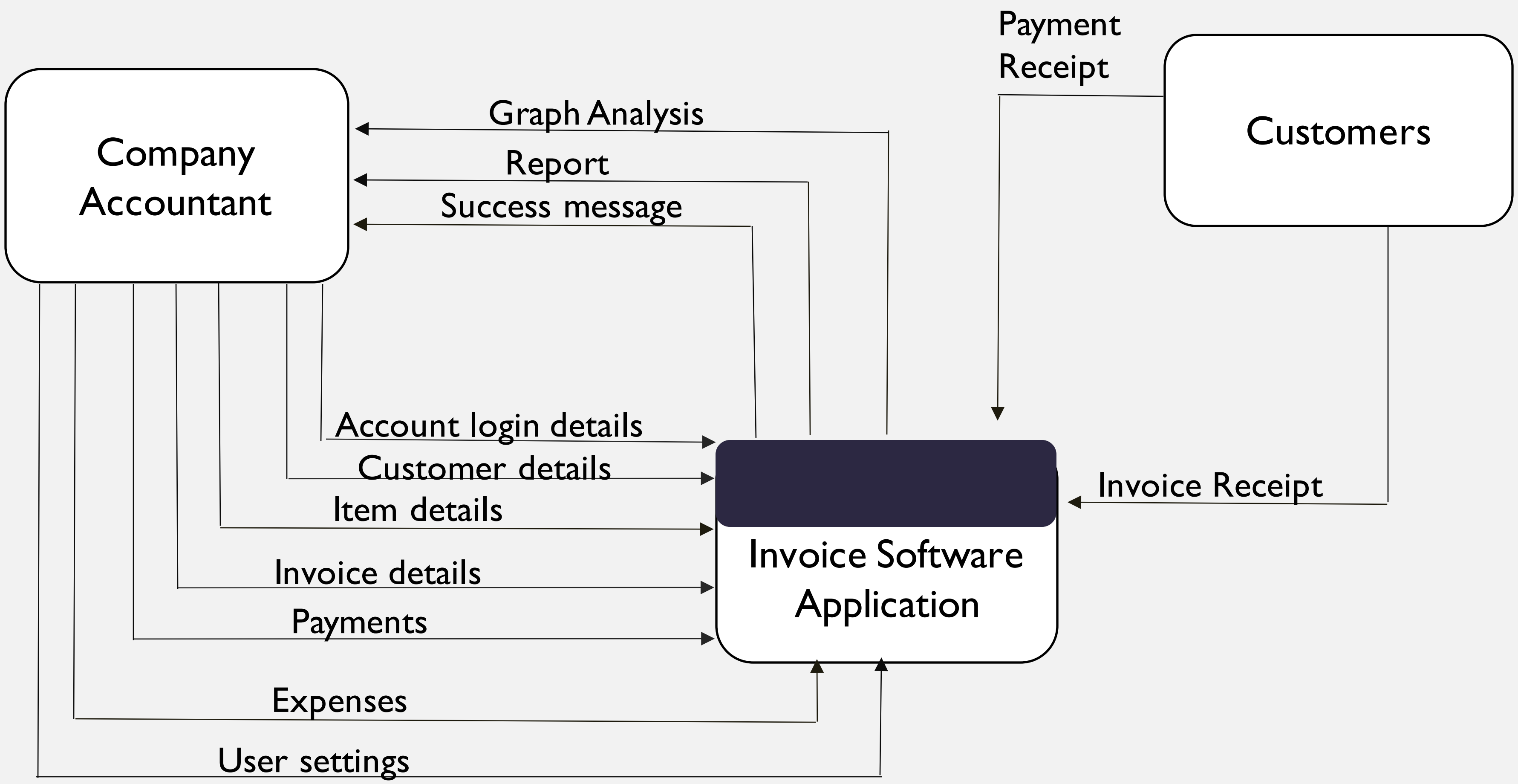
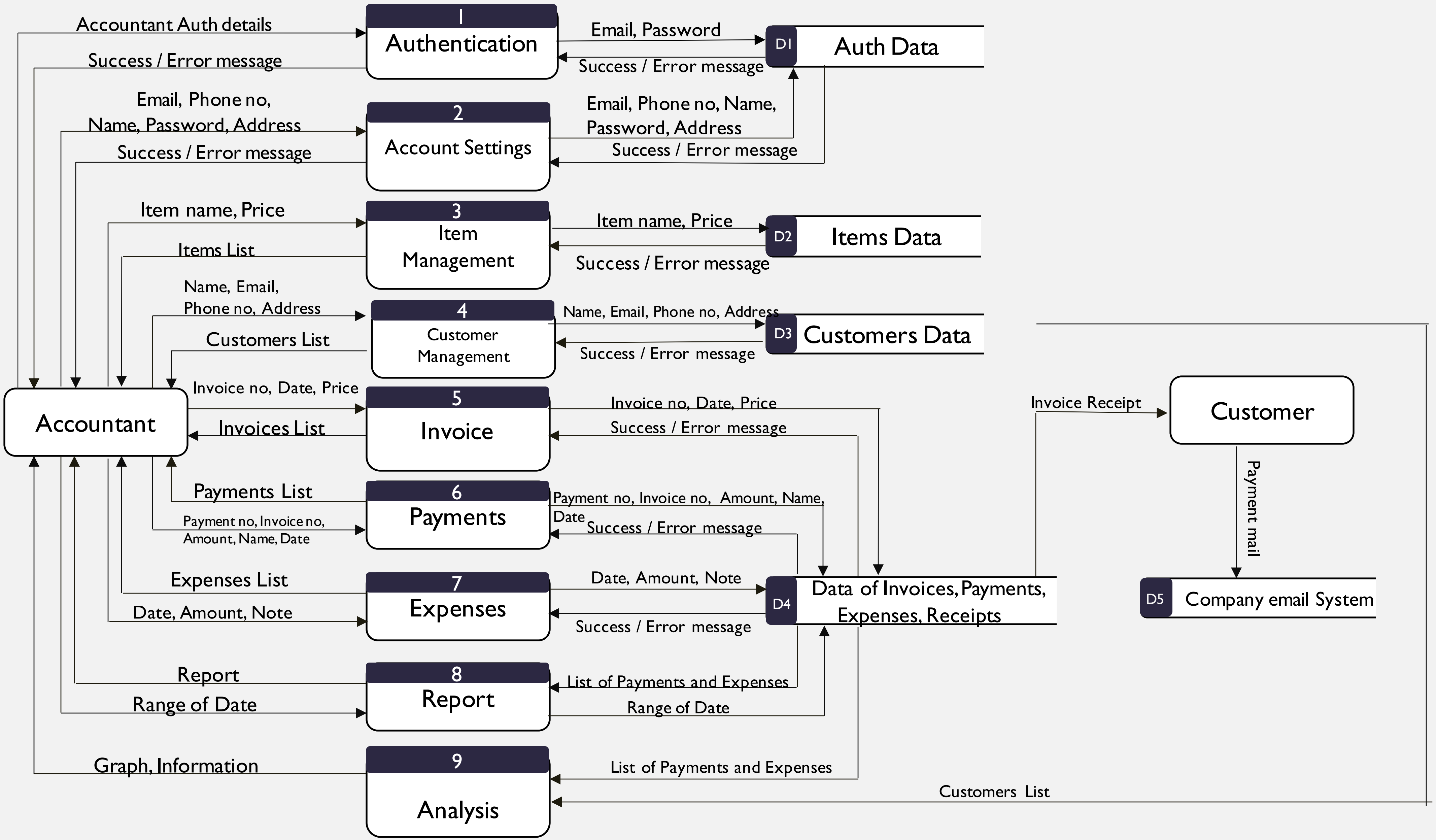
DFD FOR INVOICE SOFTWARE

- 190101028 – Bolleboina Madhumitha
- 190101030 – Chappidi Shreya
- 190101042 – Kalapati Kasvitha
- 190101056 – Mushanolla Pranathi

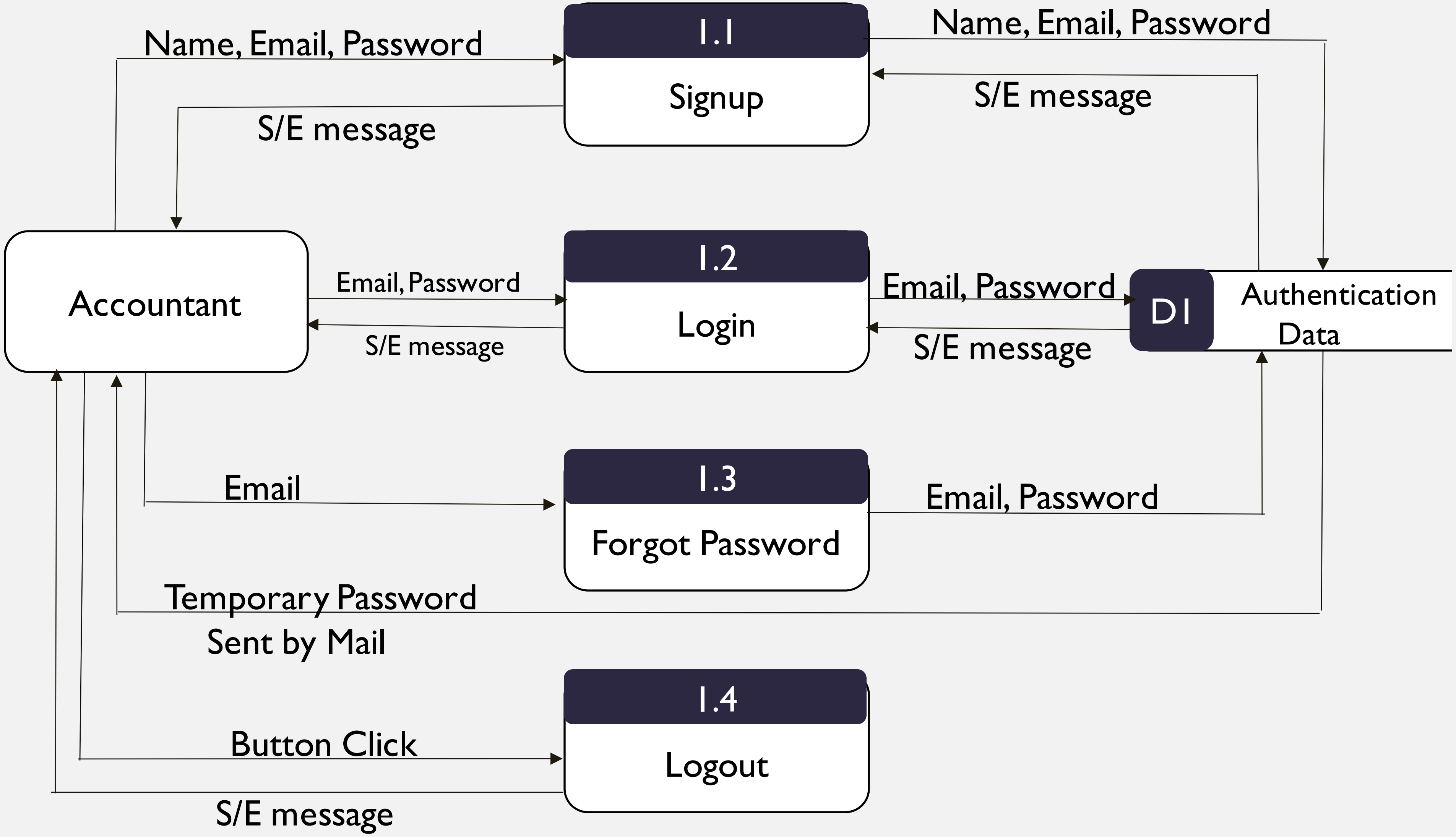
LEVEL 0


Payment
Receipt
Customers
Company
Accountant
Graph Analysis
Report
Success message
Account login details
Customer details
Item details
Invoice details
Payments
Expenses
User settings
Invoice Software
Application
Invoice Receipt

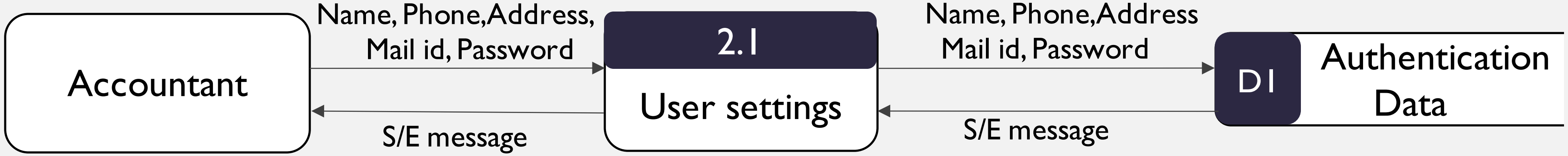
LEVEL 1


Accountant Auth details
Success / Error message
1
Authentication
Email, Password
Success / Error message
D1
Auth Data
Email, Phone no,
Name, Password, Address
Success / Error message
2
Account Settings
Email, Phone no, Name,
Password, Address
Success / Error message
Item name, Price
Items List
3
Item
Management
Item name, Price
Success / Error message
D2
Items Data
Name, Email,
Phone no, Address
Customers List
4
Customer
Management
Name, Email, Phone no, Address
Success / Error message
D3
Customers Data
Invoice no, Date, Price
Invoices List
Accountant
5
Invoice
Invoice no, Date, Price
Success / Error message
Invoice Receipt
Customer
Payment mail
Payments List
Payment no, Invoice no,
Amount, Name, Date
6
Payments
Payment no, Invoice no, Amount, Name,
Date
Success / Error message
Expenses List
Date, Amount, Note
7
Expenses
Date, Amount, Note
Success / Error message
D4
Data of Invoices, Payments,
Expenses, Receipts
D5
Company email System
Report
Range of Date
8
Report
List of Payments and Expenses
Range of Date
Graph, Information
9
Analysis
List of Payments and Expenses
Customers List

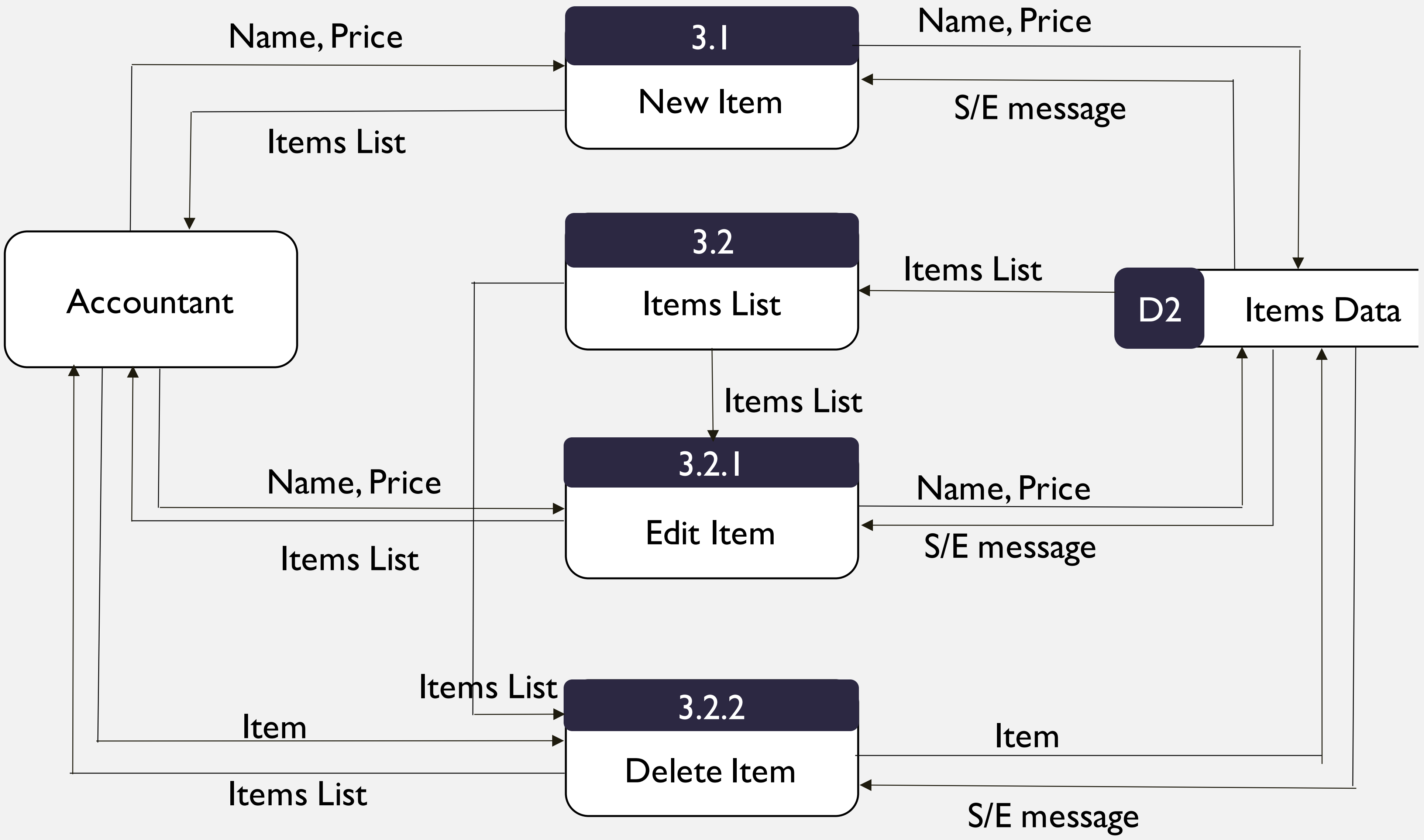
LEVEL 2 PROCESS 1



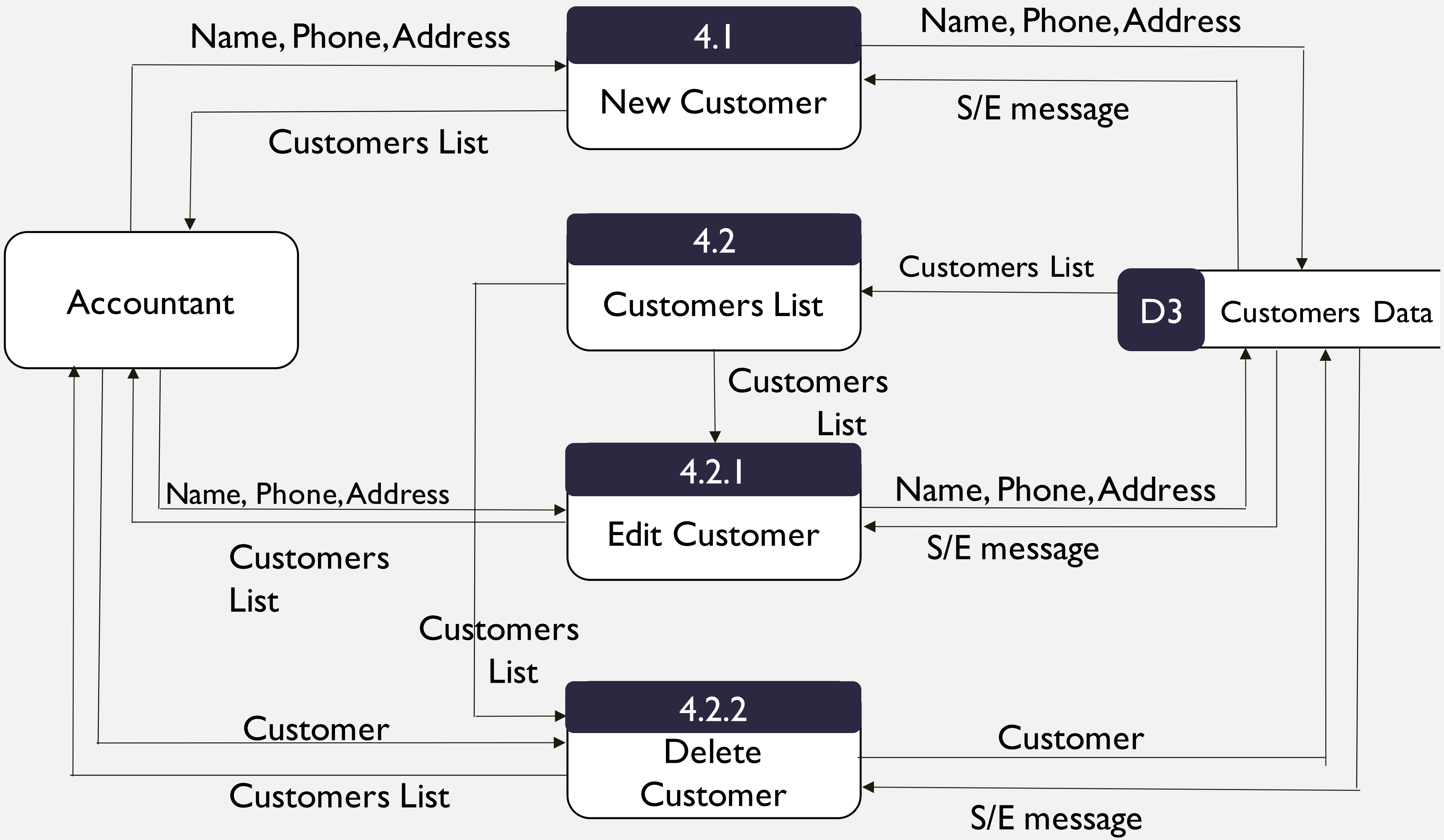
LEVEL 2 PROCESS 2



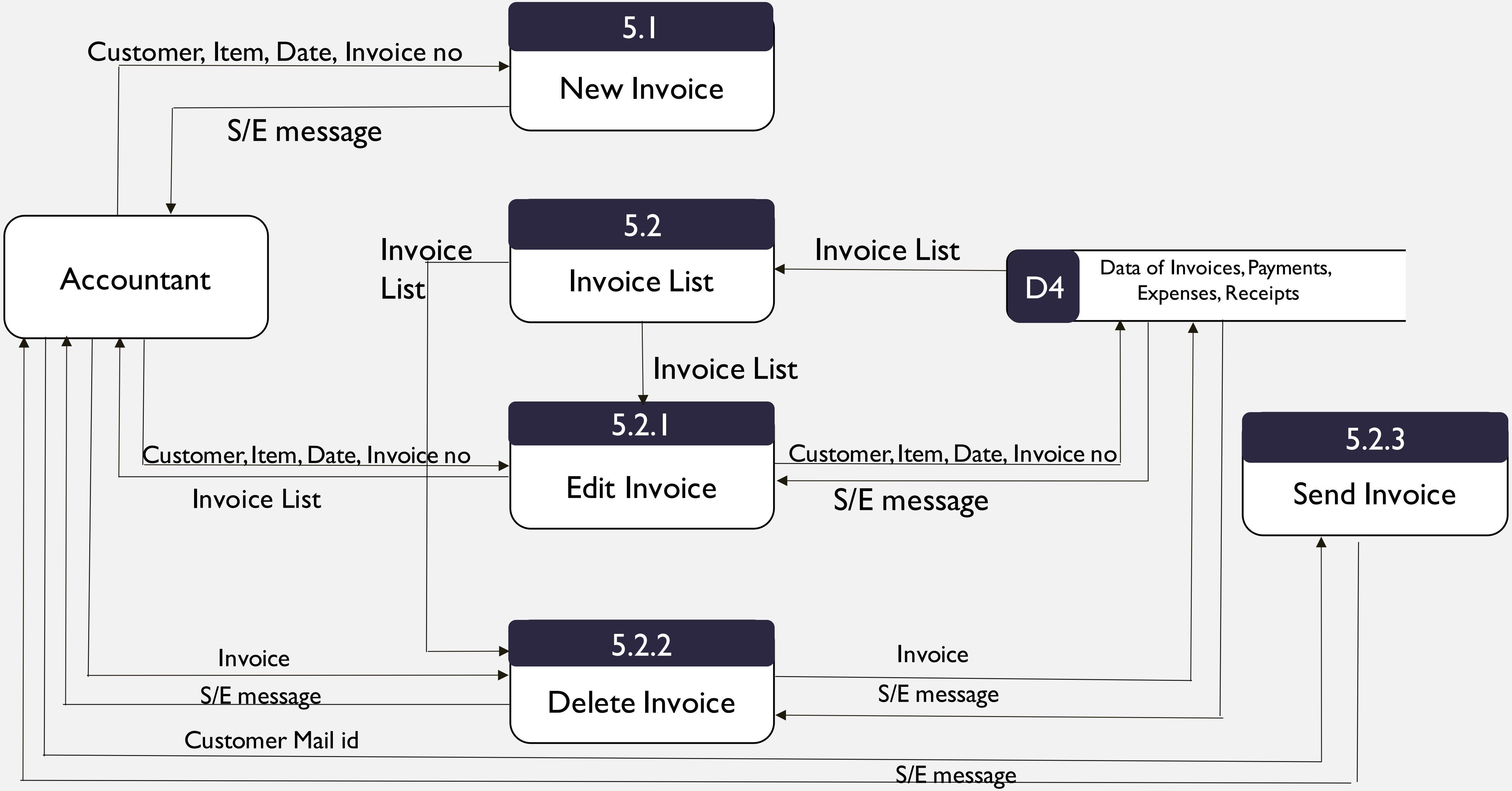
LEVEL 2 PROCESS 3



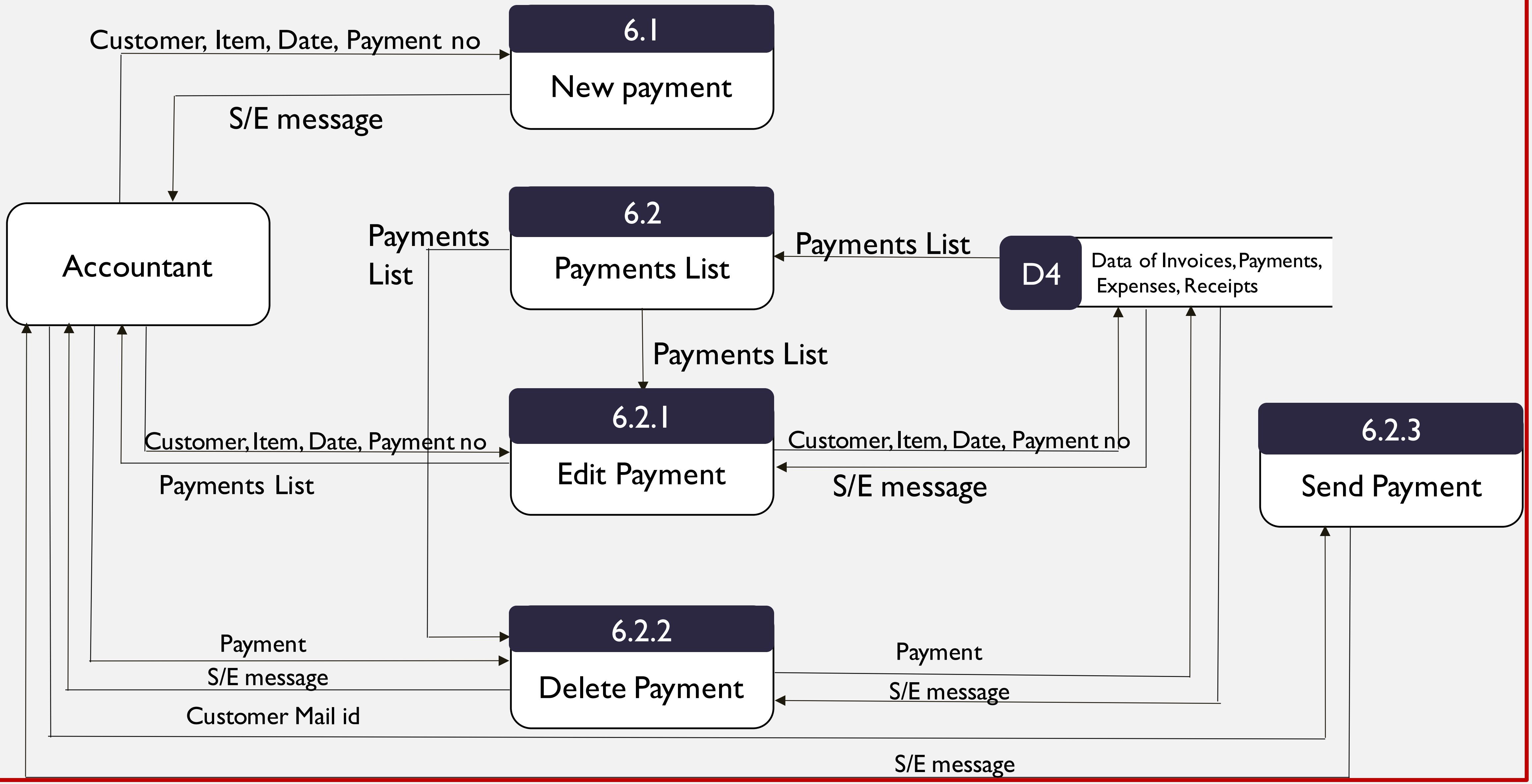
LEVEL 2 PROCESS 4



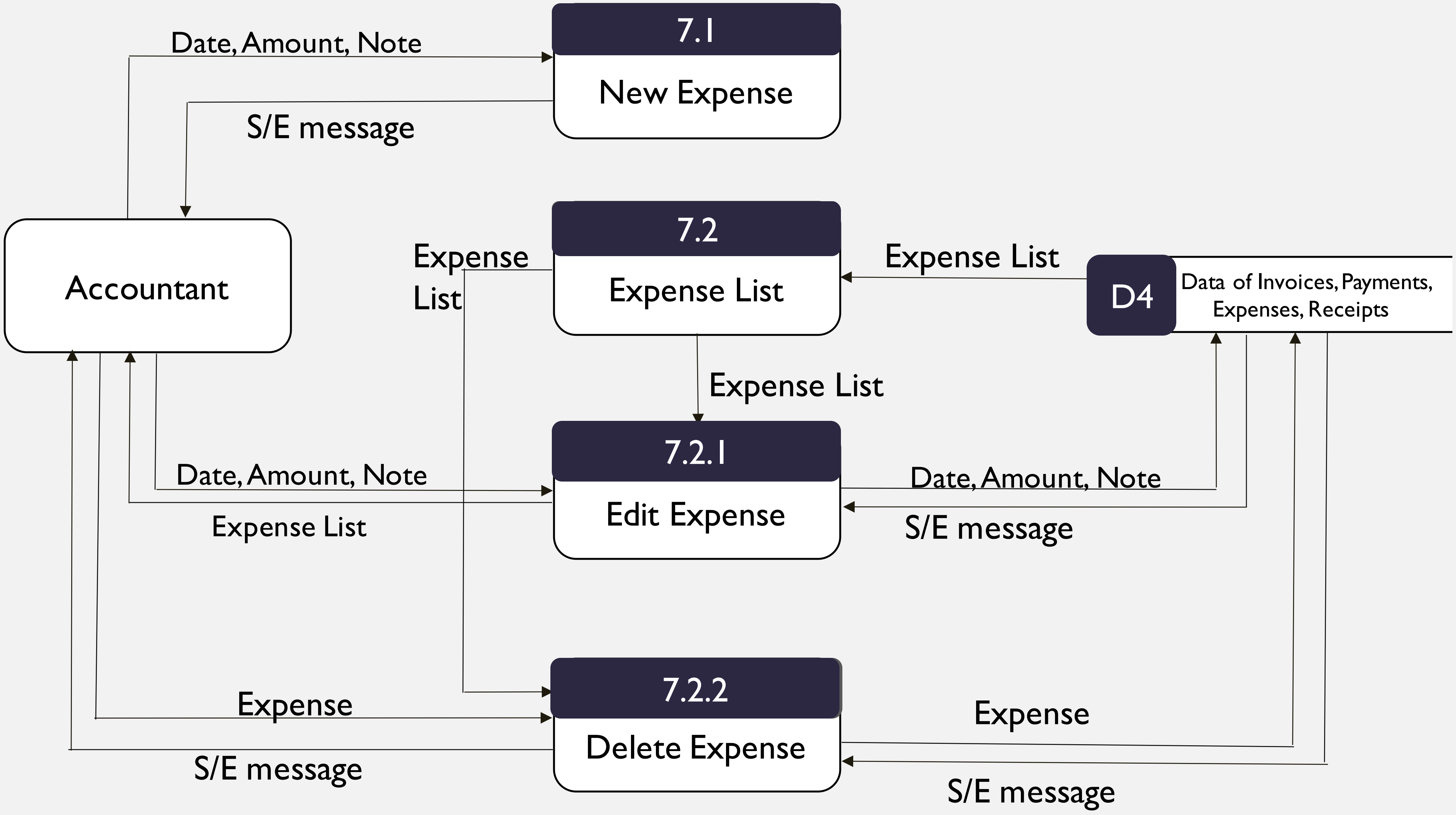
LEVEL 2 PROCESS 5



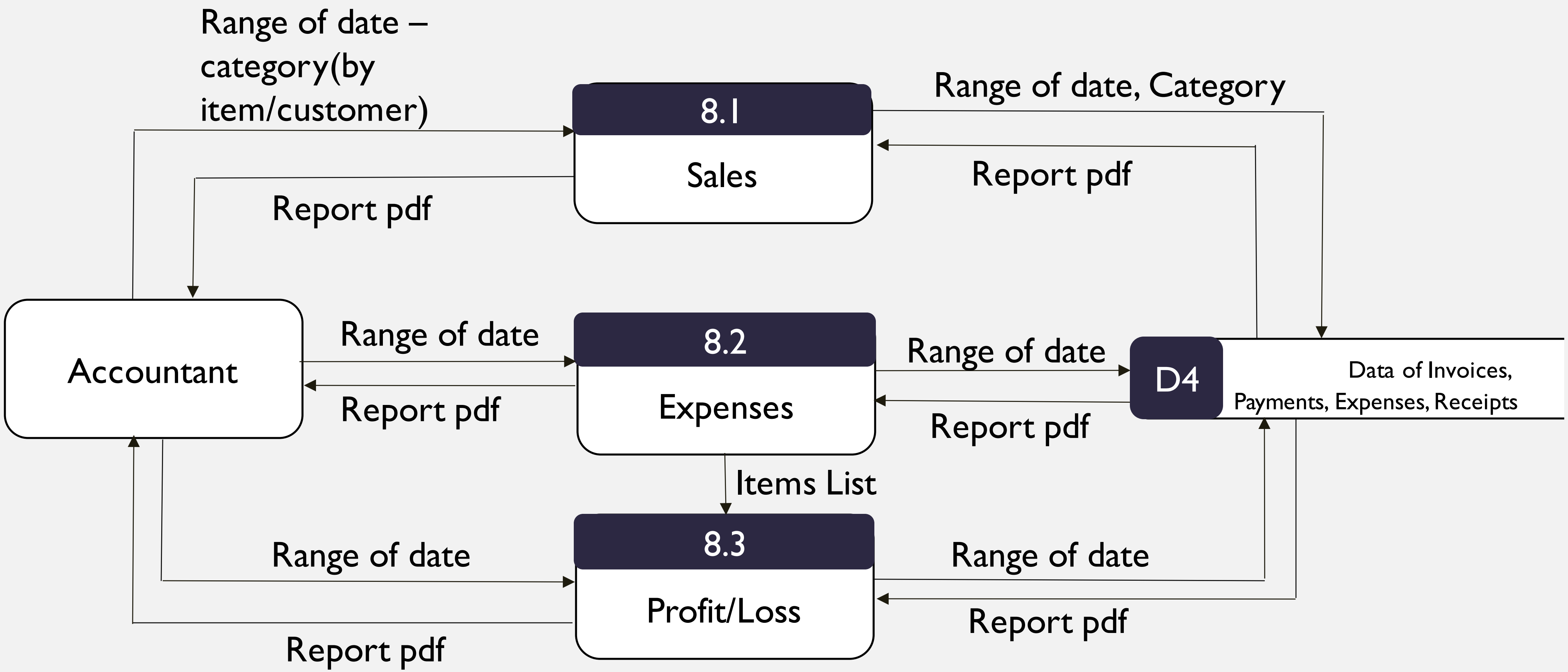
LEVEL 2 PROCESS 6



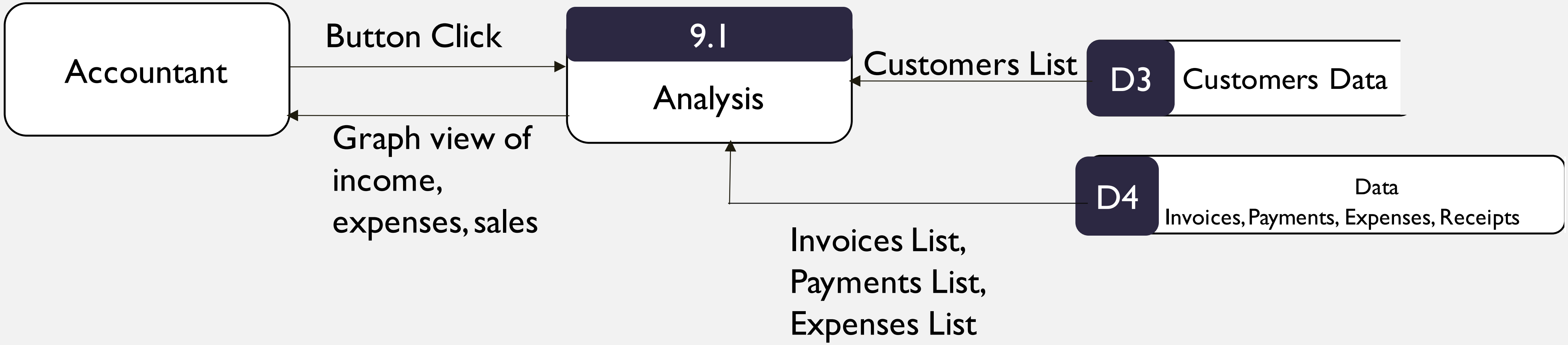
LEVEL 2 PROCESS 7



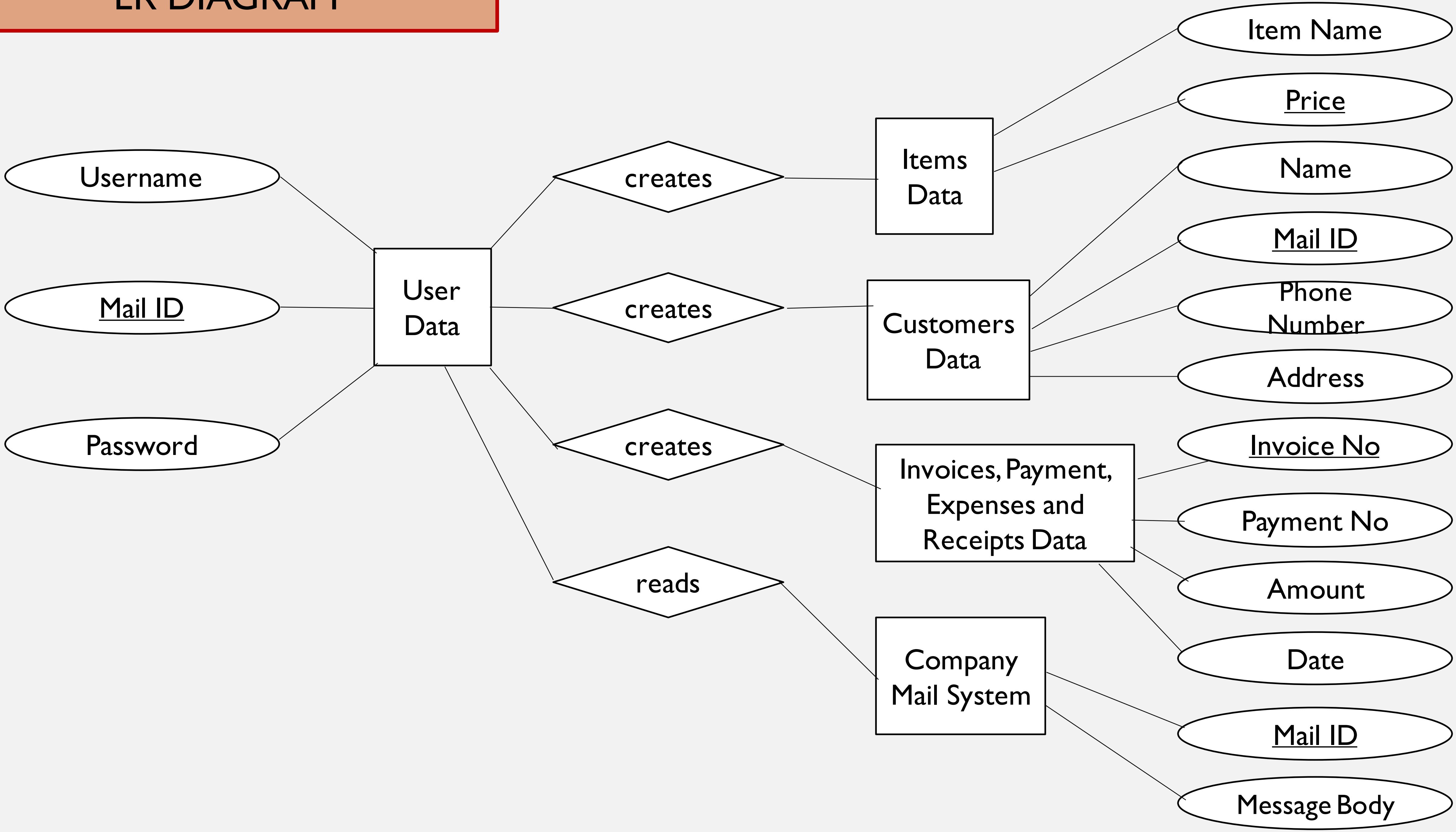
LEVEL 2 PROCESS 8



LEVEL 2 PROCESS 9



ER DIAGRAM


Username
Mail ID
Password
User Data
creates
creates
creates
reads
Items Data
Customers Data
Invoices, Payment, Expenses and Receipts Data
Company Mail System
Item Name
Price
Name
Mail ID
Phone Number
Address
Invoice No
Payment No
Amount
Date
Mail ID
Message Body

USABILITY DOCUMENT

We have highlighted the usability of our Invoice Application using the eight golden rules of Shneiderman.

1. Strive for Consistency :
People spend most of their time using other similar products and their experiences with those products set their expectations. It can frustrate our user if our design is inconsistent and not familiar to users.
a. Internal Consistency:We must maintain Consistency with inside of our software.
--- Icons : windows,mac,linux make use of the same button icon to perform the same action across all the windows. In our application we maintained the same consistency with buttons, icons. For example back button anywhere in our application means navigating back to the back page.
--- Color : In Our proposed application design we maintained uniform color consistency so that it is easier for users to keep track. Using different colors for different buttons makes user feel unpleasent. In our applicatiion we have chosen basic color theme to be purple,black,ash and all the application is designed with the same set of colours to make the application design look pleasant.

USABILITY DOCUMENT

b. External/Environmental Consistency: Follow established industry conventions. To facilitate user interaction, all functionalities of our system will cater to the established notion in the society. For example,
---Icons: The add button has a symbol of plus which can be understandable by the user.
The Delete icon has a symbol of dustbin which helps the user to identify its operation just by looking at it.

2. Design for universal usability:
Recognize the needs of diverse users and design, facilitating the transformation of content.
- For expert users we added additional features like Updating profile and graph anlaysis for the sales of the company.
- For users who are not good at remembering their passwords we designed forgot password feature
- For first time novice users our application will showcase a step by step approach on how to navigate between tabs and how to create and delete invoice were shown.
- Hence our application is designed for all kind of users keeping in mind universal usability requirements.

USABILITY DOCUMENT

3. Offer Informative Feedback:

Informing the users all the time about what's going on in the application

Human Understandable Reactions-

when the accountant clicked on any button the button will be hovered and highlighted so that user can keep track where exactly hie is clicking and where exactly the pointer of mouse is.

Continuous Feedback:

when the accountant want to download the sales reports pdfs it takes time to download if the file size is large. So inorder keep informing the user loader circle is displayed to show how much of the pdf is downloaded.

4. Design dialogues to yield closure:

a. when the accountant send a mail to the customer the application delivers a success message if the email is delivered successfully and it shows unsuccessful message if the mail is failed to deliver.

b. when the accountant update his profile or when he logged and logged out the success and error messages are displayed on the application.

USABILITY DOCUMENT

5. Offer error prevention and simple error handling:

When error occurs, users should be provided with simple, step-by-step instructions to solve the problem.

a. conflicting functionalities are not kept together in order to prevent the wrong clicks from the user.

For example, delete button and create button are kept farther.

b. When the accountant fails to login or fails to create invoices the system intimate the user about where the inputs were given wrong so that user tries again correcting his inputs.

6. Permit easy reversal of actions:

permit the reversal of actions relieves anxiety since the user knows that errors can be undone; it thus encourages exploration of unfamiliar options.

a. Back Button: This option provides user freedom to reverse the changes made by selecting the wrong option. Example: Suppose a user wants to see the invoice list but mistakenly opened payments list now the user can go back to home page by clicicking back button.

USABILITY DOCUMENT

7. Keep users in control:

Making the users the initiators of actions rather than the responders to actions. users strongly desire the sense that they are in charge of the interface and that the interface responds to their actions.

a. Confirm or Delete Message:

When the user wants to delete an invoice, payment, expense he gets the confirmation message if he really wants the item to be deleted.

b. save button:

When the user creates invoice, payment, expense he have to click on save button to create new invoice, payment, expense.

8. Reduce short term memory load:

limitation of human information processing in short term memory requires that display be kept simple. Keeping our interface consistent will help us to make our design more intuitive so our user doesn't have to call every time he/she uses the product. It's simpler for us to recognize information rather than recall it.

a. add button have plus symbol and delete have dustbin symbol .

All these visual elements are easy to recognize because they resemble real-world things that serve the same purpose.

COHESION

UML design should strive to achieve high cohesion and low coupling among its classes. That is what we have achieved in this application design too. The following are the occurrences of different types of cohesion in our design followed by the justification of how few classes in our design are tightly coupled and mostly loose coupling is seen.

Cohesion: Encapsulation enables a class to be highly cohesive i.e. their purpose is very clear. Put another way, the class does one thing, only one thing, and it does that only one thing well. Most of the classes we made have a single purpose.

1. Logical cohesion: A class exhibits logical cohesion if the tasks its methods perform are conceptually related which can be seen In authentication, login(), signup() and logout() are logically related to authentication and in invoice, editinvoice(), deleteInvoice(), sendInvoice() are logically related to invoice module.

COHESION

1. **Temporal cohesion:** Temporal cohesion is present in a class if some of its methods are executed in the same time span . This can be seen in the list of invoices class , where both deleteInvoice() and invoiceList() are performed together for deleting an invoice , and also in saveInvoice() and closeInvoice() when the user closes the invoice , saving unfinished invoice as a draft.

2. **Communicational cohesion:** A communicationally cohesive module is one which performs several functions on the same input or output data. The classes, invoices, payments and expenses works at the same time when sales class calls these classes for calculating report.

3. **Functional cohesion:** Functional cohesion can be seen in the authentication module, invoice module as well as the expenses module. The goal of the invoice module is to provide for functions that allow the user to organize his/her invoices according to their liking . For example making new invoices, new expenses, new payments . Similarly, All the functions related to creation and modification of invoice(create invoice, edit invoice, delete invoice to name a few) are grouped under the invoice module and all the functions related to authentication of users, logging them in and out of the system are grouped under the authentication module.

COUPLING

Coupling: Most Classes are lightly coupled. But tight coupling can be seen between invoice and payments , expenses, and sales , reports, where the creation of latter's object is dependent on the initial one. For example createinvoice() in Invoice class is the method that is going to create a invoice Object.

1. Data Coupling: There is minimal data coupling between modules. The data is only passed while creation of an object or while passing messages to perform a subroutine. One of the few examples of this that can be seen is between invoice list and create invoice, where the create invoice class is going to provide the data to invoice list class for displaying contents on the screen.

2. Control Coupling: Modules do not pass control information to each other. Therefore, control coupling is almost non-existent.

3. Content Coupling: This type of coupling can be seen across create invoice , edit invoice, delete invoice classes as all of them contain the listInvoice() method , the code of which is going to be the same for all of them.similar is the case for the expenses , payments class.