Software Requirement Specification Document for Automatic Recognition of Suitable Design Pattern Project

Clara Kamal, Farida Mohamed, Hashem Mohamed, Veronia Emad
Supervised By: Dr. Taraggy Mohiy and Eng. Nada Ayman

February 25, 2020

1 Introduction

According to the English dictionary, Design Pattern (DP) is defined as a formal approach of recording a generic reusable solution to a design problem in a specific environment. Software design pattern is considered as one of the most productive discoveries in the software industry due to its major role in improving the software quality. However, choosing the suitable design pattern for each design problem is considered a challenging mission that requires a good base knowledge about each DP and its functionality.

1.1 Purpose of this document

Through this Software Requirements Specification document, we aim to provide a detailed description for the proposed approach. It presents an automatic selection of the suitable design pattern depending on the user requirements using a Q/A approach through a GQM-Based Tree Model. As result, it will achieve the software engineers' task, with the most accurate results. This document is intended for the stakeholders and developers of the system to present the functional/non-functional requirements, purpose and goals of the proposed approach.

1.2 Scope of this document

Since the manual selection process of DPs is considered challenging and critical, the proposed system can be described as an assisting tool for software engineers (SWE). It will help in overcoming difficulties and misunderstandings that may come across software engineers while selecting the suitable design pattern in different scenarios. Moreover, the system is considered as a control point for

managing the back-end composition by the system administrator. This document presents the characteristics, objectives and constraints for each of the users: software engineer and system administrator.

1.3 Overview

As shown in Figure 1, the process starts with taking input from the software engineer. He inserts his objects and class then fills the form of questions concerning the DPs categories that are retrieved from the database. These form questions and user answers pass through the GQM-Based Tree Model to be used for the next phase, which is the weight calculation phase. This phase will retrieve the weight of each metric (answer) in order to calculate the weight results for each category. Hence, the highest weight value will be the selected category. The next step will retrieve the DPs questions for the selected category in the previous step then the user will start filling them. The same process of selecting the appropriate category will be repeated in order to select the suitable DP. The Highest weight weight value will be the selected DP. The last phase will be the class diagram generation. It will use the selected DP from the previous step and the objects and classes that the user inserted at the beginning to generate the formatted PUML file to be applied to PlantUML API which will generate the class diagram.

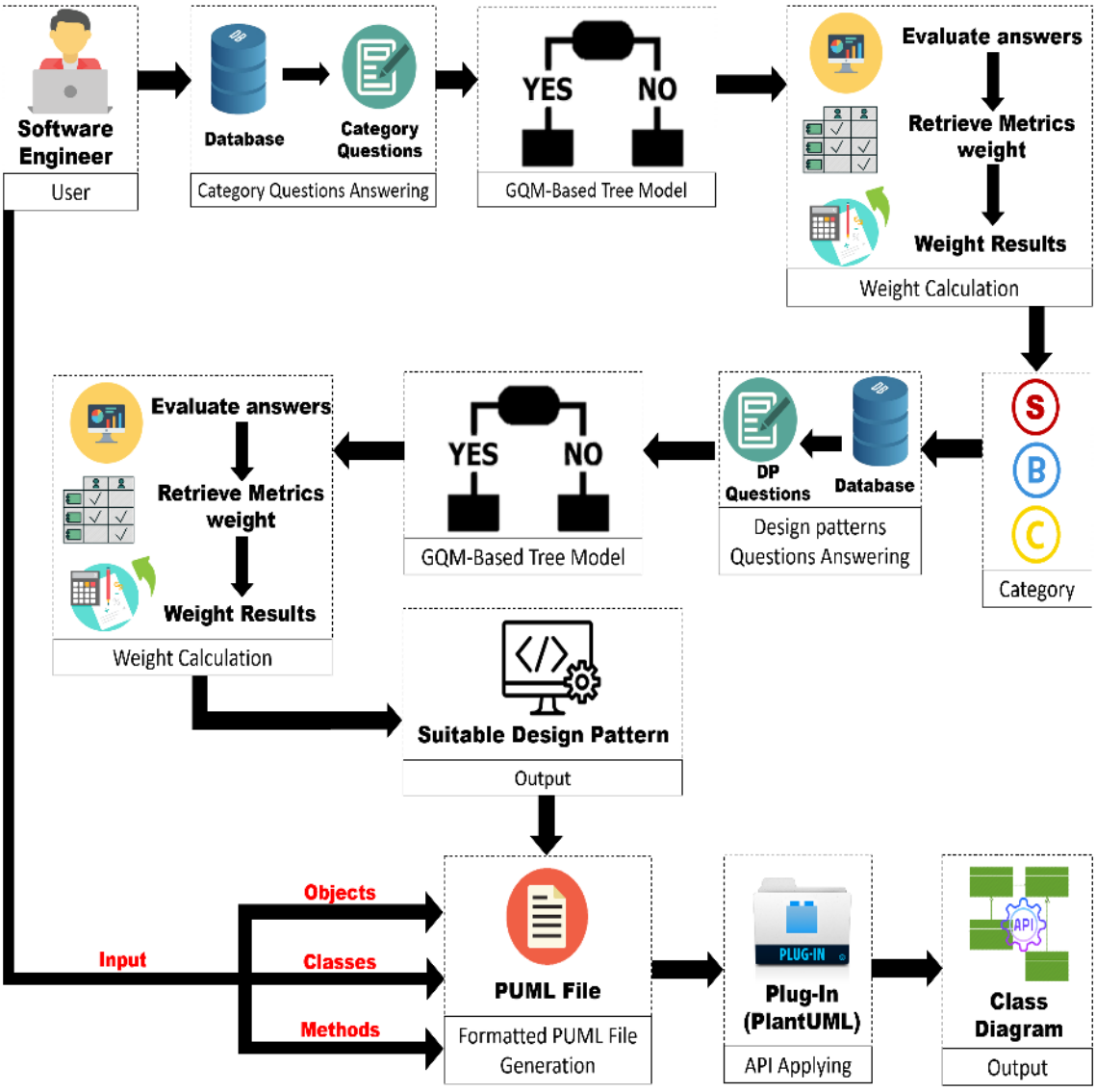


Figure 1: System Overview

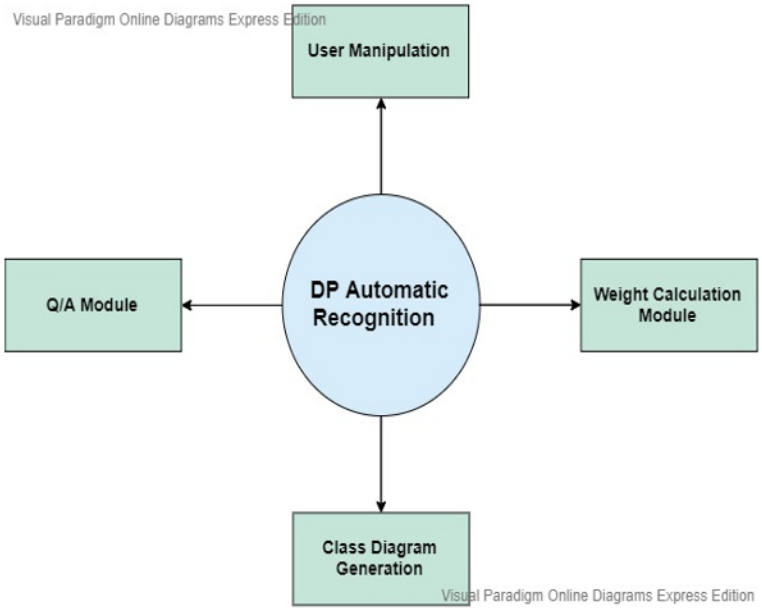


Figure 2: Context Diagram

1.4 Business Context

As the process of detecting the appropriate software design patterns for each design problem is a main concern in all software engineer's problems in nowadays, there is a an important need for an automated DPs detector. It will save time, provide more accurate results and cost less than manual detection. Moreover, our proposed automated system is aiming to reduce the confusion that may cause wrong selections and misunderstanding due to the similarities in the purposes, actions and structures of several design patterns.

2 General Description

2.1 Product Functions

Software Engineer:

- User Authentication
- View HomePage
- View Profile
- Edit Profile
- Answer Questions
- View DP Descriptions
- View Result
- Print Result
- View History
- Delete History Record
- Clear All History

System Administration:

- User Authentication
- Add Admin
- View All Users
- Delete User
- CRUD DPs
- CRUD DP Description
- CRUD DP Question
- CRUD DP Weight
- CRUD Category
- CRUD Category Description
- CRUD Category Question
- CRUD Category Weight

Weight Calculation Module:

- Retrieve User Answers
- Retrieve Metrics Weights
- Compute Final Metric Weight

2.2 Similar System Information

A recommender system [5] relies on a Goal-Question-Metric approach [2] to produce a software process pattern [1] recommendation system. The system consists of two phases: Q/A designing phase and pattern recommendation phase. The Q/A designing phase contains four steps. First, pre-processing of the process pattern library. Then, building a topic model and get text-topic probability distribution matrix by applying LDA [3] on all software process patterns scenario descriptions. Next,calculating sentences clustering using K-means algorithm [6] to probability distribution matrix and calculating topic similarity using Euclidean distance using the formula (1).

$$distance(x, y) = \sqrt{\sum_{i=1}^{k}(x_i - y_i)^2} \tag{1}$$

Finally, the designing of the questions from each cluster of pattern descriptions then assigning the correct answer to each question on every related pattern. The second phase which is the recommendation of the pattern depending on the answers, starts by recording the weight of each answer. It calculates all the matching degrees between patterns and questions then selecting the top five process patterns. The system designed 57 questions for 89 patterns. The system was evaluated by comparing it with similar systems using TF-IDF (3) and the evaluation result shows that this approach contributes a high F-score (2) which is 11.6% higher than that of the traditional TF-IDF approach. Moreover, the average precision (5) can reach 57%.

$$F - score = \frac{2 * Precision * Recall}{Recall + Precision} \tag{2}$$

$$tfidf_{i,d} = tf_{i,d} \times \log\left(\frac{N}{df_i}\right) \tag{3}$$

$$Recall = \frac{TP}{TP + FN} \tag{4}$$

$$Precision = \frac{TP}{TP + FP} \tag{5}$$

2.3 User Characteristics

The system concerns two types of users, Software Engineers and System Administrators. Each user-type needs some requirements in order to get the highest benefit from the system.

- Software Engineer:
  - Should have a moderate background about the definition of design patterns, their aim and how they are used
  - Should have a moderate background about class diagrams
  - Should have basic computer skills
  - Should be able to define all the requirements needed to give a valid answer for each question in the selection process.
- System Administrator:
  - Should have basic computer skills
  - Should have a standard knowledge about the system's functionality
  - Should have a reliable educational knowledge about each design pattern
  - Should have aware of the system's composition behind its functionality
  - Should be able to deal with and manage the database through the GUI displayed
  - Should have a moderate background about class diagrams and the class diagram representation of each design pattern

2.4 User Problem Statement

Software engineers face a difficulty in selecting a suitable design pattern for the system they are developing based on the user requirements. This may lead to the wrong selection a design pattern, which is known as Anti-Patterns [4], which makes it worse than using no DP at all. As a result, software engineers must re-design and re-implement their systems after discovering the anti-patterns in their codes, wasting time, effort and cost. For demonstration, a questionnaire was made to get the percentage of wrong design pattern selections. Thus, according to Fig.3, 72.7% of the responses showed that they already faced a problem before. Therefore, determining the suitable DP needs to be an efficient and accurate process.

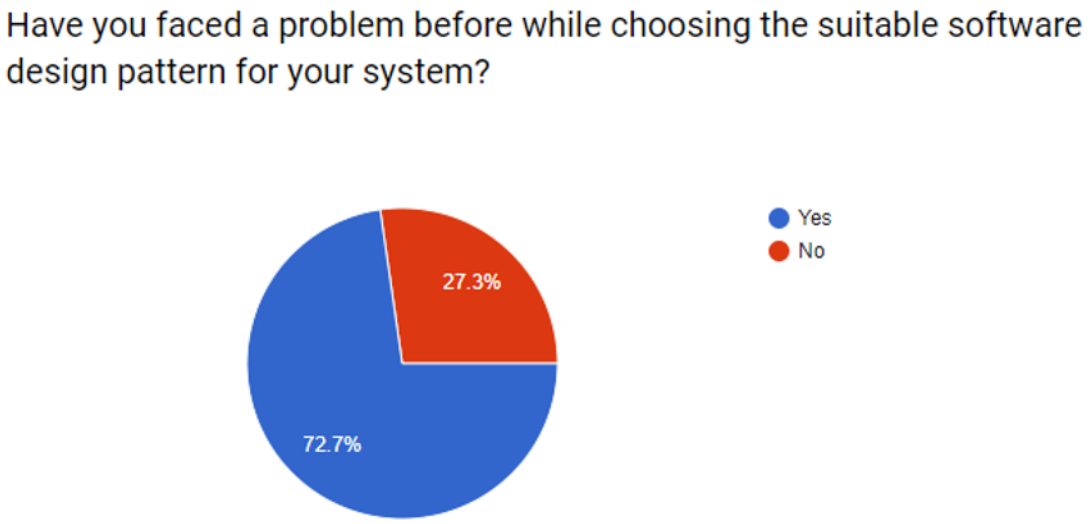


Figure 3: Statistics

2.5 User Objectives

The system is developed in a form of an assisting tool for software engineers in order to:

- Provide an automatic selection of the suitable design pattern through requirements gathering in the form of Question/Answer.
- Generate a UML class diagram based on the design pattern selected and user requirements
- Provide a description for the design pattern selected at the end of the process to help the software engineer understand the reason of the selection

Moreover, the system will provide the administrator with a friendly GUI that allows him to maintain and update the selection structure by adding, updating and removing questions of the requirements gathering form.

2.6 General Constraints

- The system is applicable only for PCs and laptops.
- Java JDK must be installed.

3 Functional Requirements

3.1 Software Engineer

3.1.1 LogIn

Name	Log In
Description	It's an authentication level that needs to be passed in order to let the users access their accounts and use the system.
Input	The user's Email and Password
Output	Homepage will be displayed
Requirements	User must be logged out
Action	User must be registered(email and password exists)
Source	GUI
Destination	Database
Pre-Condition	Email and password are found
Post-Condition	User is logged in
Side Effect	None

3.1.2 Log out

Name	LogOut
Description	User logs out from his account
Input	None
Output	Message to confirm logging out then display log in screen
Requirements	User must be logged in
Action	None
Source	GUI
Destination	GUI
Pre-Condition	None
Post-Condition	User is logged out
Side Effect	None

3.1.3 Register

Name	Register
Description	User registers a new account to the system
Input	Email,First Name,Last Name, Password
Output	A message to confirm adding the user account
Requirements	User must be a software engineer
Action	Check that Email exists and not repeated
Source	GUI
Destination	Database
Pre-Condition	None
Post-Condition	User is added in the database with user-type software engineer
Side Effect	None

3.1.4 Update Info

Name	Update Info
Description	User can edit them his information
Input	User ID
Output	A message to confirm updating the user information
Requirements	User must be logged in
Action	Check if the user id exists
Source	GUI
Destination	Database
Pre-Condition	Get user information from GUI
Post-Condition	User data is updated
Side Effect	None

3.1.5 Profile

Name	Profile
Description	Display the user profile
Input	User ID
Output	User Profile is displayed
Requirements	User must be logged in
Action	Check if user id exists
Source	Database
Destination	GUI
Pre-Condition	None
Post-Condition	This user's information are retrieved
Side Effect	None

3.1.6 View History

Name	ViewHistory
Description	Display All Previous User's Recommendation Processes
Input	User ID
Output	History Records are displayed
Requirements	User must be logged in
Action	Check if the user have previous recommendation processes
Source	Database
Destination	GUI
Pre-Condition	None
Post-Condition	All history records for this user id are retrieved
Side Effect	None

3.1.7 View Homepage

Name	ViewHomepage
Description	Display homepage to user
Input	None
Output	Homepage is displayed
Requirements	User must be logged in
Action	None
Source	Database
Destination	GUI
Pre-Condition	None
Post-Condition	Data retrieved from database
Side Effect	None

3.1.8 View Result

Name	ViewResult
Description	User view the final output of the recommended design pattern and the generated class diagram
Input	Process ID
Output	Result Form is displayed
Requirements	User must be logged in as software engineer
Action	None
Source	Database
Destination	GUI
Pre-Condition	All questions must be answered
Post-Condition	Result is retrieved and displayed
Side Effect	None

3.1.9 View DP Descriptions

Name	ViewDPDescriptions
Description	User view all DP names and descriptions
Input	None
Output	DP names and descriptions
Requirements	User must be logged in
Action	Check if DP descriptions is not empty
Source	Database
Destination	GUI
Pre-Condition	None
Post-Condition	DP names and descriptions are retrieved from database
Side Effect	None

3.1.10 Print Result

Name	PrintResult
Description	Convert the result form to PDF form to be printed
Input	result
Output	PDF file
Requirements	the existence of a result
Action	takes the result if existed and convert it to PDF and display it to be dowbloaded.
Source	database
Destination	GUI
Pre-Condition	ViewResult
Post-Condition	None
Side Effect	None

3.1.11 Delete History

Name	DeleteHistory
Description	User delete a record from his previous recommendation processes
Input	History record ID and user ID
Output	A message to confirm that the record was deleted
Requirements	User must be logged in
Action	Check if user have previous history
Source	GUI
Destination	Database
Pre-Condition	None
Post-Condition	History record is deleted from database with specific id
Side Effect	None

3.1.12 Delete All History

Name	DeleteAllHistory
Description	User deletes all his previous recommendation processes records
Input	User ID
Output	A message to confirm that all records were deleted
Requirements	User must be logged in
Action	Check if user have previous history
Source	GUI
Destination	Database
Pre-Condition	None
Post-Condition	All History records are deleted from database
Side Effect	None

3.1.13 Generate Questions

Name	Generate Questions
Description	Questions are retrieved from database to user in order to let him start answering them
Input	Question ID, Number of Question
Output	The Questions Form
Requirements	User must be logged in as a software engineer
Action	If a Question is not answered, the next question won't be displayed. else, it will display next question and take the answer input to calculate the total weight.
Source	User Input
Destination	Database
Pre-Condition	Insert Category
Post-Condition	Calculate Weight
Side Effect	None

3.1.14 Calculate Result

Name	Calculate Result
Description	It calculates the weight of the user's answers to help him find the Category with the highest score of weights. It retrieves the answers weights from the database then calculate their sum.
Input	Question ID, Answer ID, Answer Weight
Output	The Result
Requirements	All the Questions must be answered by the user and all the Answers must have weights in the database.
Action	If an Answer doesn't have weight the weight won't be calculated. else, it will display the Result and show the user the suitable Category according to his answers.
Source	Database
Destination	GUI
Pre-Condition	Answer Questions
Post-Condition	View Result
Side Effect	None

3.2 System Administrator

3.2.1 Add User

Name	Add User
Description	Create new admin account to be able to choose their usertype to avoid letting anyone from choosing their own usertype to avoid conflicting permissions
Input	Firstname, lastname, emailaddress, password, usertype
Output	username and password given to the faculty member
Requirements	it'll take the information from the inputs fields
Action	the AddUser form will first ask the admin about the usertype then based on his choice it'll show or hide the needed input fields according to this choice to fill them.
Source	inputs from the admin
Destination	database
Pre-Condition	logged in as admin
Post-Condition	generate the user id of the admin
Side Effect	none

3.2.2 View User

Name	View User
Description	it will retrieve the user's info from the database
Input	user id
Output	the selected User is displayed
Requirements	the id of the user must exist in the database
Action	The admin will search for a user with id so the function will read the info of the user.
Source	Database
Destination	GUI
Pre-Condition	create user
Post-Condition	Edit User and Delete User
Side Effect	none

3.2.3 View All Users

Name	View All Users
Description	it will retrieve the users' info from the database
Input	none
Output	users info
Requirements	none
Action	The admin will display all the info of all users.
Source	Database
Destination	GUI
Pre-Condition	create user
Post-Condition	Edit User and Delete User
Side Effect	none

3.2.4 View All Categories

Name	View All Categories
Description	it will retrieve the Categories' info from the database
Input	none
Output	Categories info
Requirements	none
Action	The admin will display all the info of all Categories.
Source	Database
Destination	GUI
Pre-Condition	create Category
Post-Condition	Edit Category and Delete Category
Side Effect	none

3.2.5 View All Design Patterns

Name	View All Design Patterns
Description	it will retrieve the Design Patterns' info from the database
Input	none
Output	Design Patterns info
Requirements	none
Action	The admin will display all the info of all Design Patterns.
Source	Database
Destination	GUI
Pre-Condition	create Design Pattern
Post-Condition	Edit Design Pattern and Delete Design Pattern
Side Effect	none

3.2.6 Delete User

Name	Delete User
Description	It will update the user's is_deleted column in the database to 1.
Input	user id
Output	none
Requirements	the id of the user must exist in the database
Action	It will take the id of the user and deletes it.
Source	database
Destination	database
Pre-Condition	Create User and View User
Post-Condition	none
Side Effect	none

3.2.7 Encryption/Decryption

Name	Encryption/Decryption
Description	To make password encrypted with encrypted keys
Input	password
Output	Encrypted password
Requirements	there must be a password taken as an input
Action	It encrypts password and reads password from table users in database and decrypt it into it's original form and then give this password
Source	password from table users in database
Destination	Database
Pre-Condition	Readable password
Post-Condition	hidden password
Side Effect	none

3.2.8 Insert Category

Name	Insert Category
Description	Create new design pattern category.
Input	Category name, Category description, Category questions, Category question's yes weight, Category question's no weight
Output	none
Requirements	none
Action	It will insert a new category to the database by setting all the info needed for each category which is the Category name, Category description, Category questions, Category question's yes weight, Category question's no weight if it doesn't already exist or the input is not null.
Source	user input
Destination	database
Pre-Condition	none
Post-Condition	edit category, view category, delete category, insert design pattern, edit design pattern, view design pattern, delete design pattern, view all categories, view all design patterns.
Side Effect	none

3.2.9 Edit Category

Name	Edit Category
Description	Update an existing design pattern category.
Input	Category name, Category description, Category questions, Category question's yes weight, Category question's no weight
Output	none
Requirements	the id of the selected Category must exist in the database
Action	The admin will search for a Category with id so the function will read the info of the Category selected if the id already exists and lets the admin edits it.
Source	user input
Destination	database
Pre-Condition	insert category, view category
Post-Condition	delete category, insert design pattern, edit design pattern, view design pattern, delete design pattern, view all categories, view all design patterns.
Side Effect	none

3.2.10 View Category

Name	View Category
Description	View the chosen design pattern category and display all its info.
Input	Category id
Output	the selected Category is displayed
Requirements	the id of the selected Category selected must exist in the database
Action	The admin will search for a Category with id so the function will display the info of the Category if the id already exists.
Source	database
Destination	GUI
Pre-Condition	insert category
Post-Condition	edit category, delete category, insert design pattern, edit design pattern, view design pattern, delete design pattern.
Side Effect	none

3.2.11 Delete Category

Name	Delete Category
Description	It will update the Category's is_deleted column in the database to 1.
Input	Category id
Output	none
Requirements	the id of the Category selected must exist in the database
Action	It will take the id of the Category and deletes it if the id already exists.
Source	database
Destination	database
Pre-Condition	Insert Category and View Category
Post-Condition	none
Side Effect	none

3.2.12 Insert Design Pattern

Name	Insert Design Pattern
Description	Create new design pattern category.
Input	Category ID, Design Pattern name, Design Pattern description, Design Pattern questions, Design Pattern question's yes weight, Design Pattern question's no weight
Output	none
Requirements	none
Action	It will insert a new Design Pattern to the database by setting all the info needed for each category which is the Design Pattern name, Design Pattern description, Design Pattern questions, Design Pattern question's yes weight, Design Pattern question's no weight if it doesn't already exist or the input is not null.
Source	user input
Destination	database
Pre-Condition	none
Post-Condition	edit Design Pattern, view Design Pattern, delete Design Pattern, view all design patterns.
Side Effect	none

3.2.13 Edit Design Pattern

Name	Edit Design Pattern
Description	Update an existing design pattern category.
Input	Category ID,Design Pattern name, Design Pattern description, Design Pattern questions, Design Pattern question's yes weight, Design Pattern question's no weight
Output	none
Requirements	the id of the selected Design Pattern must exist in the database
Action	The admin will search for a Design Pattern with id so the function will read the info of the Design Pattern selected if the id already exists and lets the admin edits it.
Source	user input
Destination	database
Pre-Condition	insert Design Pattern, view Design Pattern
Post-Condition	delete Design Pattern, view all Design Patterns.
Side Effect	none

3.2.14 View Design Pattern

Name	View Design Pattern
Description	View the chosen design pattern and display all its info.
Input	Design Pattern id
Output	the selected Design Pattern is displayed
Requirements	the id of the selected Design Pattern selected must exist in the database
Action	The admin will search for a Design Pattern with id so the function will display the info of the Design Pattern if the id already exists.
Source	database
Destination	GUI
Pre-Condition	insert Design Pattern
Post-Condition	edit Design Pattern, delete Design Pattern, view all Design Patterns.
Side Effect	none

3.2.15 Delete Design Pattern

Name	Delete Design Pattern
Description	It will update the Design Pattern's is_deleted column in the database to 1.
Input	Design Pattern id
Output	none
Requirements	the id of the Design Pattern selected must exist in the database
Action	It will take the id of the Design Pattern and deletes it if the id already exists.
Source	database
Destination	database
Pre-Condition	Insert Design Pattern and View Design Pattern
Post-Condition	none
Side Effect	none

4 Interface Requirements

4.1 User Interfaces

4.1.1 GUI

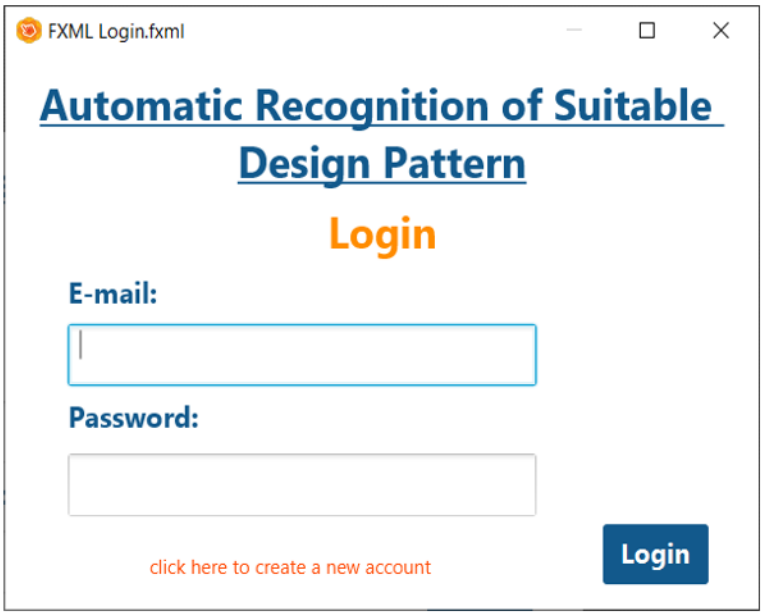


Figure 4: Log In Form

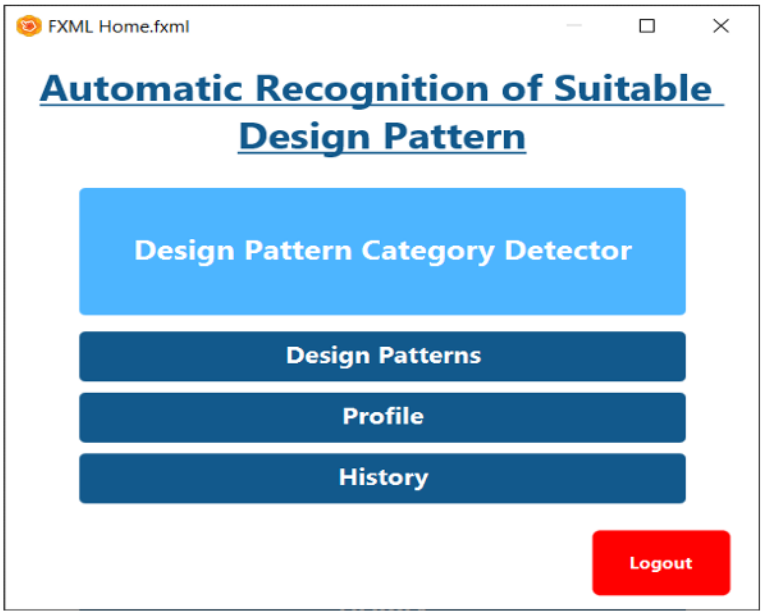


Figure 5: Homepage

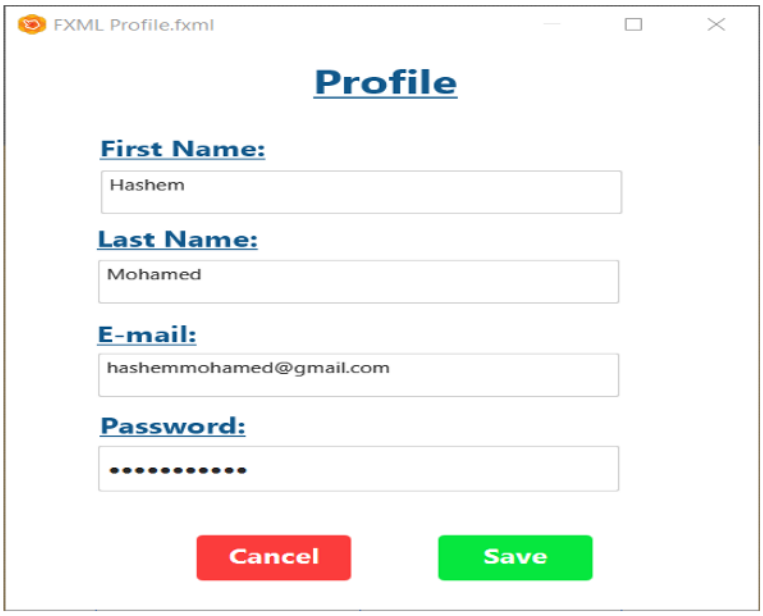


Figure 6: User Profile

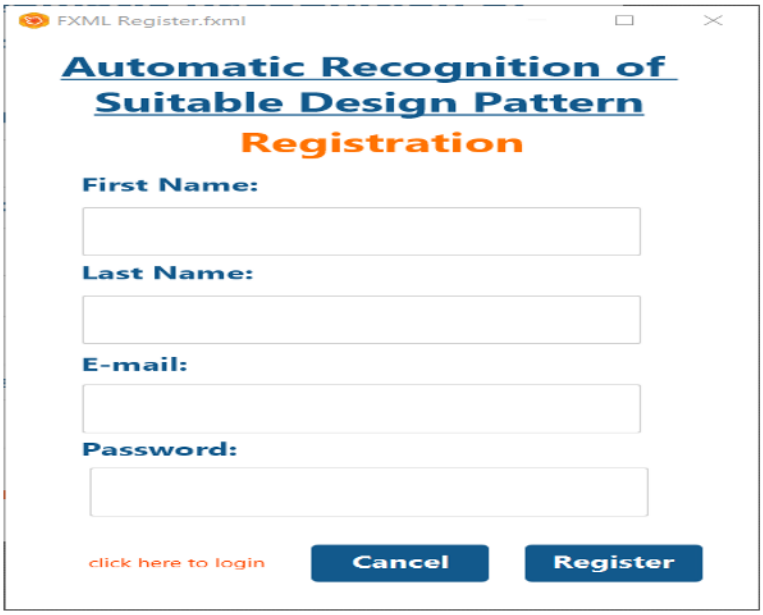


Figure 7: Register Form

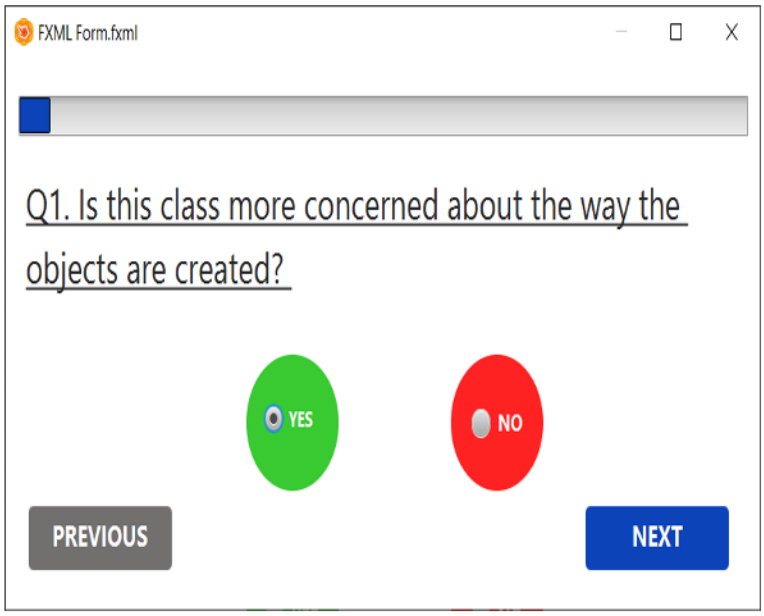


Figure 8: Form(1)

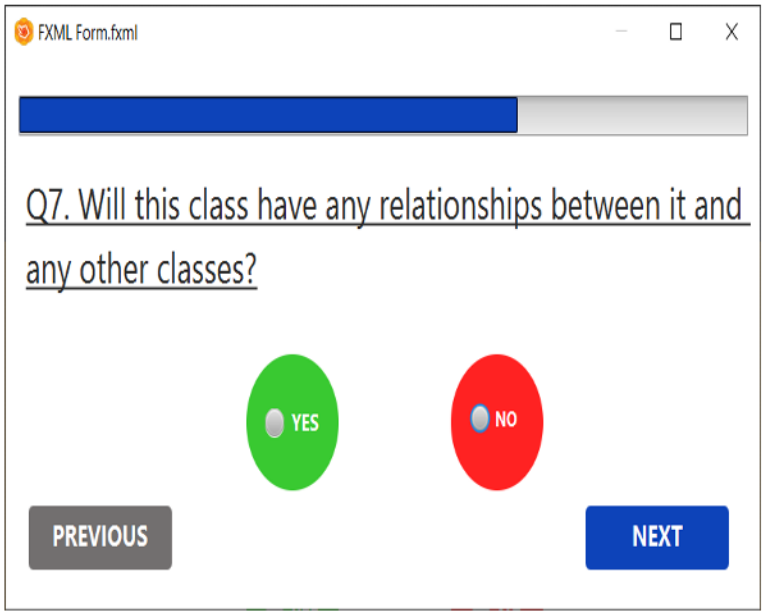


Figure 9: Form(2)

4.1.2 CLI

Git :

- git init, to initialize a new repository.
- git clone, to use an existing repository.
- git add, to add files.
- git commit -a, to commit all new files added and any changes made on existing files.
- git pull, to fetch the files in the main repository and merge to the local working copy.
- git push, to push committed changes to the main repository.

4.1.3 API

- JavaFx
- PlantUML
- Google Login
- Apache PDFBox

4.1.4 Diagnostics or ROM

N/A

4.2 Hardware Interfaces

N/A

4.3 Communications Interfaces

N/A

4.4 Software Interfaces

NetBeans will be the development environment for the system's code.

5 Performance Requirements

The system shall be able to retrieve all questions of the three categories and 23 DPs stored in the database, process all user answers then generate the result of the recommended DP. It should operate in a high speed and high recognition accuracy.

6 Design Constraints

6.1 Standards Compliance

- 64-bit operating system, x64 based processor.
- PC with 4.00 GB of RAM. (Minimum).

6.2 Hardware Limitations

Since the system is a desktop application, it won't need any hardware device except a PC or a laptop to run on.

7 Other non-functional attributes

7.1 Security

- Each user-type is restricted to some specific permissions and actions on the system to prevent accidental or malicious access.
- User's sensitive information such as passwords should never be displayed on screen. It should be encrypted using special characters.
- Each record in the database should be encrypted so if any data changed, the changed row can be detected easily.
- The system should be locked for 10 minutes in case of more than 5 successive attempts of logging in with wrong email or password.
- The system's database should be accessed only by system administrators.
- The system must be able to differentiate between the different user-types logging in attempts.
- The system must verify each email entered through the registration process.
- When the user exits the system, the user's account is logged out

7.2 Maintainability

Maintainability of the system will be achieved by using MVC design pattern that divides the system into three modules to separate the data handling in each module. The system should provide a straight-forward friendly GUI that allows the system administrator to add, update and delete any of the accessible database information easily such as the questions in the tree structure behind the design pattern selection process.

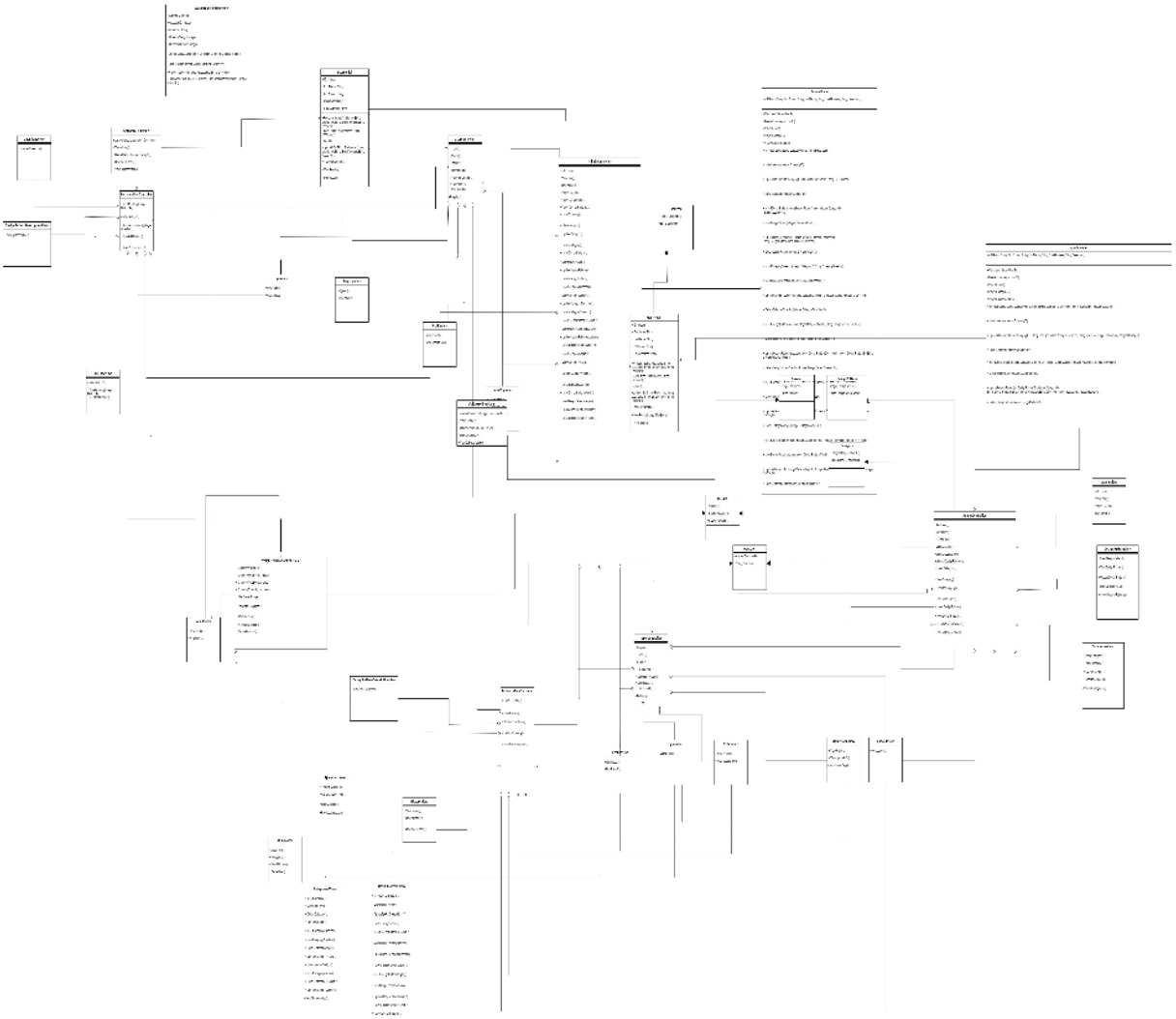
7.3 Portability

Since the system is implemented using Java, it provides source code portability. Java is platform independent since it can be written once and run anywhere. Therefore, the system is portable since it can run on Windows, Mac, Linux, etc.

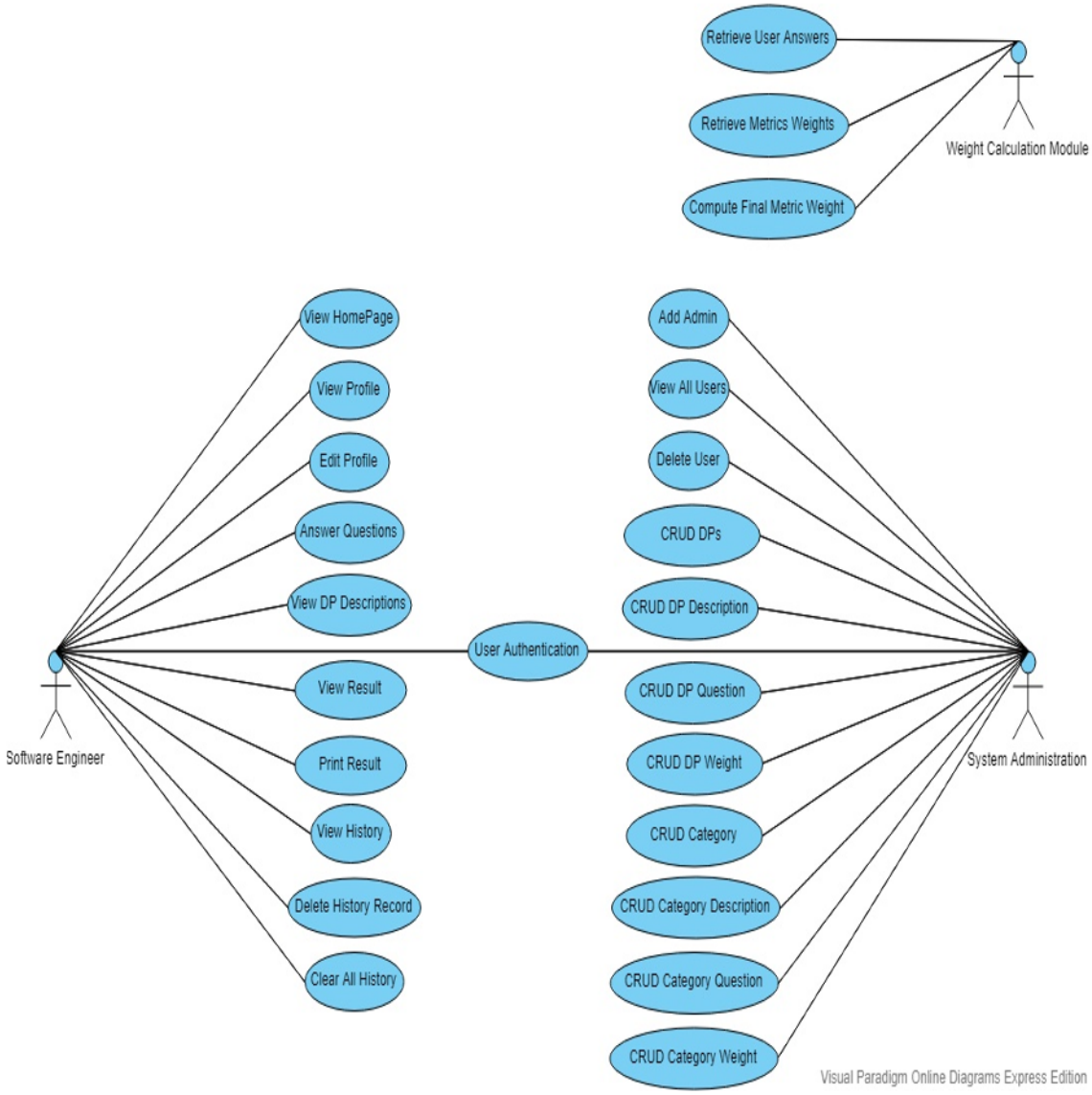
7.4 Extensibility

The system is easy to be extended and updated with the minimum effect by applying EAV data model. The system administrator can add new questions and update or delete any of the existing questions without affect the tree structure behind the design pattern selection process.

8 Preliminary Object-Oriented Domain Analysis



9 Operational Scenarios



9.1 Use Case: User Authentication

9.1.1 Log-in

Actors: Software Engineer, System Administrator
Description: The user will be able to log-in to his account
Data: User's email and password
Stimulus: User Command issued by the Software Engineer or System Administrator
Response: User will be logged in
Comments: The user must be registered

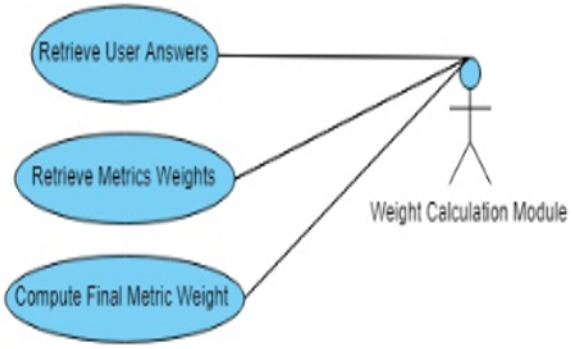
9.1.2 Log-out

Actors: Software Engineer, System Administrator
Description: The user will be able to log-out from his account
Data: None
Stimulus: User Command issued by the Software Engineer or System Administrator
Response: User will be logged out
Comments: User must be logged in

9.1.3 Register

Actors: Software Engineer
Description: The user will be able to register a new account
Data: User's information (email,name,password,birth-date)
Stimulus: User Command issued by the Software Engineer
Response: Confirmation message that a verification email has been sent
Comments: None

9.2 Weight Calculation Module



9.2.1 Use Case: Retrieve User Answers

Actors: Weight Calculation Module
Description: The answers submitted by the user are retrieved from the database
Data: User ID, Form ID
Stimulus: System Command
Response: Answers are inserted into an array
Comments: The user mush submit the answers of the form

9.2.2 Use Case: Retrieve Metrics Weights

Actors: Weight Calculation Module
Description: The weight of each user answer is retrieved from the database
Data: Answers' array
Stimulus: System Command
Response: Weights are inserted in an array
Comments: Perform after (Retrieve User Answers) use case

9.2.3 Use Case: Compute Final Metric Weight

Actors: Weight Calculation Module
Description: All the weights of user answers are added together
Data: Weights' array
Stimulus: System Command
Response: The final weight result is computed
Comments: Perform after (Retrieve Metrics Weights) use case

9.3 Software Engineer

9.3.1 Use Case: View HomePage

Actors: Software Engineer
Description: SWE will view the system's homepage
Data: None
Stimulus: User Command issued by the Software Engineer
Response: Homepage is displayed
Comments: SWE must be logged in

9.3.2 Use Case: View Profile

Actors: Software Engineer
Description: SWE will view his own profile
Data: User profile's information from the database
Stimulus: User Command issued by the Software Engineer
Response: User profile is displayed
Comments: SWE must be logged in

9.3.3 Use Case: Edit Profile

Actors: Software Engineer
Description: SWE will update his profile information
Data: User ID, an array of the changed profile values
Stimulus: User Command issued by the Software Engineer
Response: The changed values are saved in the database
Comments: SWE must be logged in

9.3.4 Use Case: Answer Questions

Actors: Software Engineer
Description:SWE will answer the view the questions form to answer it
Data: Questions from database
Stimulus: User Command issued by the Software Engineer
Response: Questions Form is displayed
Comments: SWE must be logged in

9.3.5 Use Case: View DP Descriptions

Actors: Software Engineer
Description: SWE will view all DP descriptions
Data: DP descriptions from the database
Stimulus: User Command issued by the Software Engineer
Response: DP descriptions are displayed
Comments: SWE must be logged in

9.3.6 Use Case: View Result

Actors: Software Engineer
Description: The final result of the recommended design pattern and the generated class diagram is displayed to the user after answering the form questions
Data: Highest weight value
Stimulus: User Command issued by the Software Engineer
Response: Result will be displayed
Comments: SWE must be logged in, (Compute Final Metric Weight) use case must be performed first

9.3.7 Use Case: Print Result

Actors: Software Engineer
Description: The final result of the recommended design pattern and the generated class diagram are displayed in a PDF form to be printed
Data: The final result of the recommended design pattern and the generated class diagram
Stimulus: User Command issued by the Software Engineer
Response: A PDF file is displayed
Comments: SWE must be logged in, SWE must answer all form questions and reach the final result

9.3.8 Use Case: View History

Actors: Software Engineer
Description: SWE will view the history of any previous recommendation processes done by him **Data:** All previous recommendation process for this user retrieved from database **Stimulus:** User Command issued by the Software Engineer
Response: All History records are displayed **Comments:** SWE must be logged in

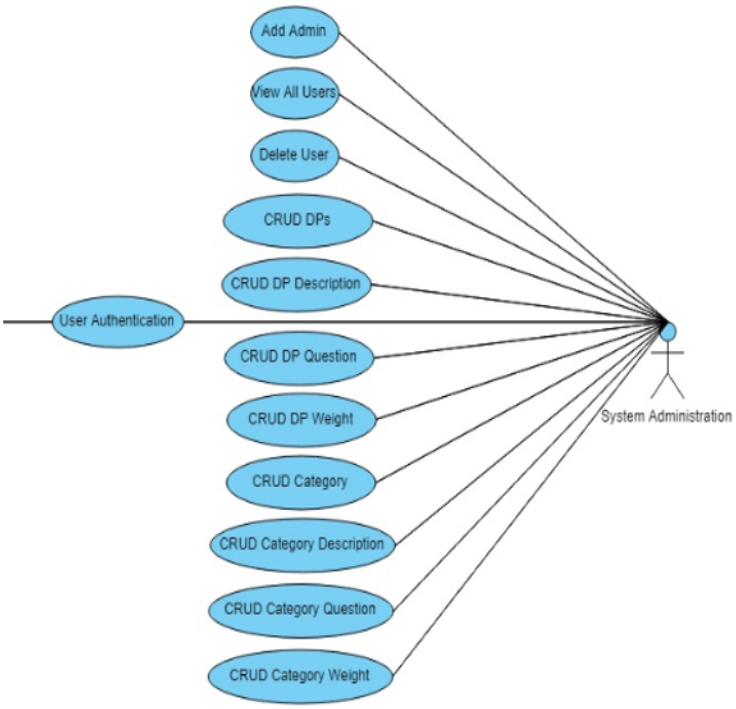
9.3.9 Use Case: Delete History Record

Actors: Software Engineer
Description: SWE will delete one record from previous recommendation processes done by him
Data: The history record id of the record to be deleted
Stimulus: User Command issued by the Software Engineer
Response: Confirmation message that the record was deleted
Comments: SWE must be logged in

9.3.10 Use Case: Clear All History

Actors: Software Engineer
Description: SWE will delete all previous records recommendation processes done by him
Data: None
Stimulus: User Command issued by the Software Engineer
Response: Confirmation message that all records were deleted
Comments: SWE must be logged in

9.4 System Administration



9.4.1 Use Case: Add Admin

Actors: System Administration
Description: The admin will register new admin by adding his information
Data: New admin's information
Stimulus: User Command issued by the Admin
Response: Confirmation message that the admin was added to the database
Comments: Admin must be logged in

9.4.2 Use Case: View All Users

Actors: System Administration
Description: The admin will view all users registered to the system
Data: All users' information retrieved from database
Stimulus: User Command issued by the Admin
Response: All users are displayed in a tabular form
Comments: Admin must be logged in

9.4.3 Use Case: Delete User

Actors: System Administration
Description: The admin will delete a registered user from database
Data: User id that is to be deleted
Stimulus: User Command issued by the Admin
Response: Confirmation message that the admin was deleted from database
Comments: Admin must be logged in

9.4.4 Use Case: CRUD DPs

Actors: System Administration
Description: The admin will create,read,update and delete design patterns to/from the database
Data: (Read operation: All DPs data retrieved from database), (Update: The DP id that is to be updated and retrieve its data then add the updated data), (Delete: The DP id that is to be deleted)
Stimulus: User Command issued by the Admin
Response: Confirmation message and display result in case of reading
Comments: Admin must be logged in

9.4.5 Use Case: CRUD DP Description

Actors: System Administration
Description: The admin will create,read,update and delete design pattern descriptions to/from the database
Data: (Read operation: All DP descriptions retrieved from database), (Update: The DP description id that is to be updated and retrieve its data then add the updated data), (Delete: The DP description id that is to be deleted)
Stimulus: User Command issued by the Admin
Response: Confirmation message and display result in case of reading
Comments: Admin must be logged in

9.4.6 Use Case: CRUD DP Question

Actors: System Administration
Description: The admin will create,read,update and delete design pattern questions to/from the database
Data: (Read operation: All DP questions retrieved from database), (Update: The DP question id that is to be updated and retrieve its data then add the updated data), (Delete: The DP question id that is to be deleted)
Stimulus: User Command issued by the Admin
Response: Confirmation message and display result in case of reading
Comments: Admin must be logged in

9.4.7 Use Case: CRUD DP Weight

Actors: System Administration
Description: The admin will create,read,update and delete weights for the questions to/from the database
Data: (Read operation: All DP weights retrieved from database), (Update: The DP weight id that is to be updated and retrieve its data then add the updated data), (Delete: The DP weights id that is to be deleted)
Stimulus: User Command issued by the Admin
Response: Confirmation message and display result in case of reading
Comments: Admin must be logged in

9.4.8 Use Case: CRUD Category

Actors: System Administration
Description: The admin will create,read,update and delete design pattern categories to/from the database
Data: (Read operation: All categories retrieved from database), (Update: The category id that is to be updated and retrieve its data then add the updated data), (Delete: The category id that is to be deleted)
Stimulus: User Command issued by the Admin
Response: Confirmation message and display result in case of reading
Comments: Admin must be logged in

9.4.9 Use Case: CRUD Category Description

Actors: System Administration
Description: The admin will create,read,update and delete design pattern categories' descriptions to/from the database
Data: (Read operation: All categories descriptions retrieved from database), (Update: The category description id that is to be updated and retrieve its data then add the updated data), (Delete: The category description id that is to be deleted)
Stimulus: User Command issued by the Admin
Response: Confirmation message and display result in case of reading
Comments: Admin must be logged in

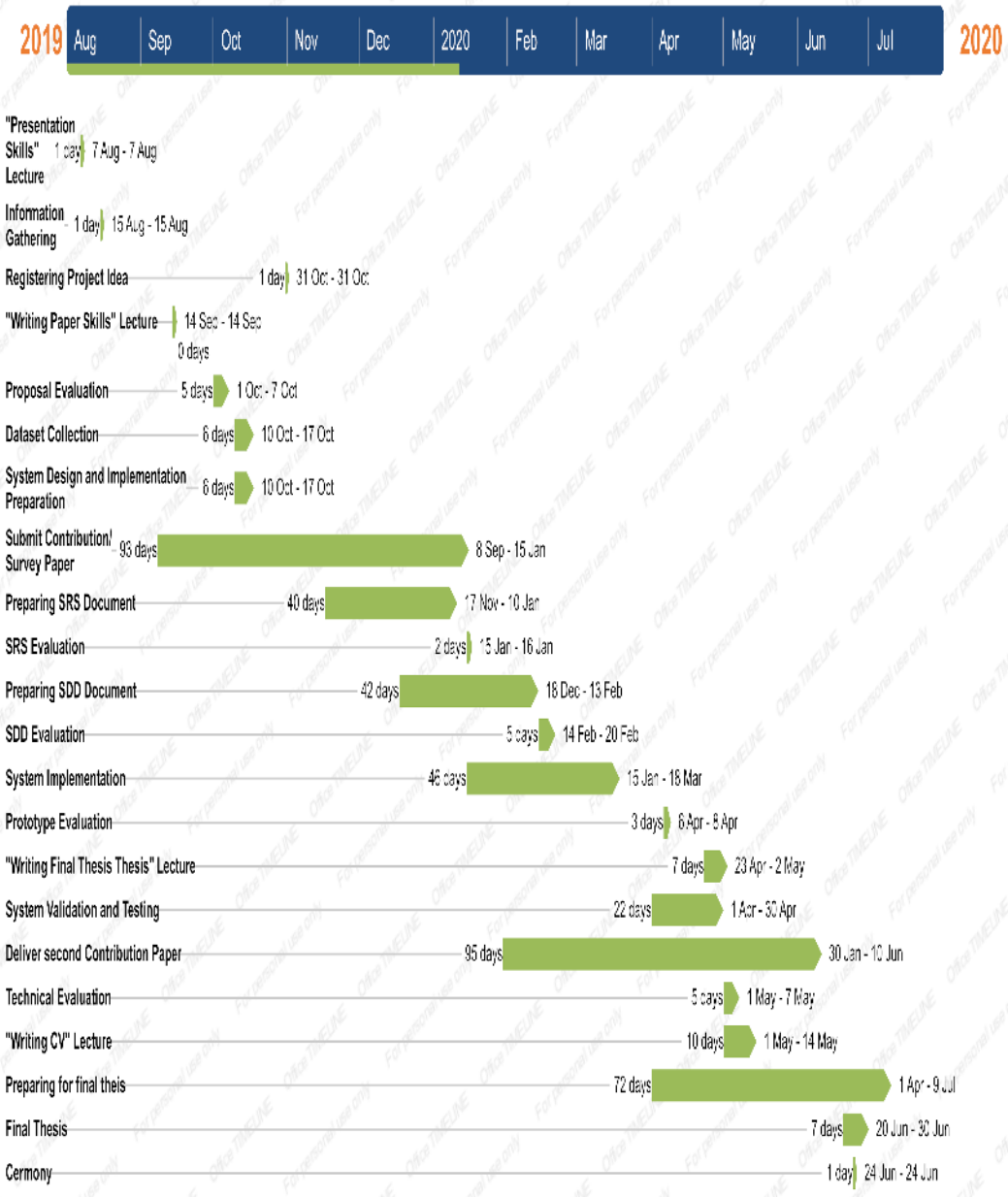
9.4.10 Use Case: CRUD Category Question

Actors: System Administration
Description: The admin will create,read,update and delete design pattern categories' questions to/from the database
Data: (Read operation: All category questions retrieved from database), (Update: The category question id that is to be updated and retrieve its data then add the updated data), (Delete: The category question id that is to be deleted)
Stimulus: User Command issued by the Admin
Response: Confirmation message and display result in case of reading
Comments: Admin must be logged in

9.4.11 Use Case: CRUD Category Weight

Actors: System Administration
Description: The admin will create,read,update and delete weights of categories' questions to/from the database
Data: (Read operation: All category weights retrieved from database), (Update: The category weight id that is to be updated and retrieve its data then add the updated data), (Delete: The category weight id that is to be deleted)
Stimulus: User Command issued by the Admin
Response: Confirmation message and display result in case of reading
Comments: Admin must be logged in

10 Preliminary Schedule Adjusted



11 Preliminary Budget Adjusted

64-bit Laptop with a cost starting from 4,499.00 EGP

12 Appendices

12.1 Definitions, Acronyms, Abbreviations

- GUI:Graphical user interface
- GQM: 3-layered approach that identifies goals, questions and metrics in the form of a tree. Goals are identified at the top, questions needed to reach these goals in the middle and at last the metrics that are the answers of the questions.
- API: Application Programming Interface
- CLI: Command Line Interface
- NetBeans: An integrated development environment for Java
- Git: a distributed version-control system for tracking changes in source code during software development.
- MySQL: An open-source relational database management system. SQL stands for Structured Query Language.
- UML: Unified Modeling Language
- PUML: PlantUML
- Software Engineer: The person responsible for the development of a software based on the principles of software engineering.
- System Administrator: The person responsible for running the system.
- MVC: Model-View-Controller design pattern
- EAV: Entity-Attribute-Value data model
- LDA: Latent Dirichlet Allocation
- Q/A: Question/Answer
- TF-IDF: Term Frequency-Inverse Document Frequency
- SWE: Software engineer

12.2 Collected material

Tables [1],[2] and [3] introduces a part of our generated dictionary, that was extracted by analyzing and researching different academic resources [7] [8]. This dictionary helps us in constructing the base of the requirements gathering process. It contains the name, intent, function, objective, disadvantage, when to use and category of each design pattern in a tabular form.

Table 1: Builder Design Pattern

Function	To split the construction of a complex object from its representation, in order to create different representation in the same construction process.
Intent	Analyze a complex representation, establish one of several targets.
Objective	(1)Construct complicated objects (2)To construct different fixed objects using the exact object building process.
Disadvantage	The code's overall complexity increases as the pattern needs multiple new classes to be developed.
When to use	(1) The algorithm for establishing a complex object should be independent of the parts that make up the object and how they're assembled. (2)The structuring process must allow different representations for the object that's built.
Category	Creational

Table 2: Singleton Design Pattern

Function	To enclose a global resource.
Intent	Make sure the only one instance is created for a class with a global access point to it.
Objective	Prevent creating multiple instances for a specific class.
Disadvantage	(1) Singleton client code is difficult to be unit tested, Needs a specialized treatment in the case of multi-threading to prevent the creation of many singleton objects. (2) May cover bad code designs, Breaks the rule of "Single Responsibility" principle.
When to use	Creating a single instance that can be accessed by all clients, Providing a harsh control on global variables.
Category	Creational

Table 3: Strategy Design Pattern

Function	To extract different algorithms in separate classes and create a generic class that connects them.
Intent	Allows initializing a collection of strategies/algorithms then encapsulate each of them and allow them to be interchangeable.
Objective	Prevent large and complex un-maintainable codes as a result of adding/editing multiple functionalities/strategies.
Disadvantage	Can be replaced with modern programming languages that make the same functionality of the design pattern without complicating the code.
When to use	(1) Switching between different algorithms during runtime. (2) Implementing multiple similar classes that differ only in their behaviour. (3) Covering some complex implementation details from the client.
Category	Behavioral

13 Approach

Unlike other approaches, this approach starts by classifying the category of the design pattern, which are Creational, Structural and Behavioral, using different sets of questions that define these categories. The Figure 10 represents the GQM based tree model. It is a combination between GQM tree and decision tree. This tree consists of 12 levels of design patterns categories' questions. Each question has two possible answers which are yes and no. Each question has different yes and no weights. The weight is calculated depending on the sum of answers' weight of each question. The proposed tree model includes sequence of questions, the software developer needs to answer these questions that's designed upon the definitions of the three categories of the 23 design patterns: Creational, Structural and Behavioral. Based on the answers, the flow of the tree model will lead to the corresponding design pattern category. The developer's requirements are extracted from the questions' answers. The weights of the answers are calculated, and accordingly the system decides for the suitable design pattern category.

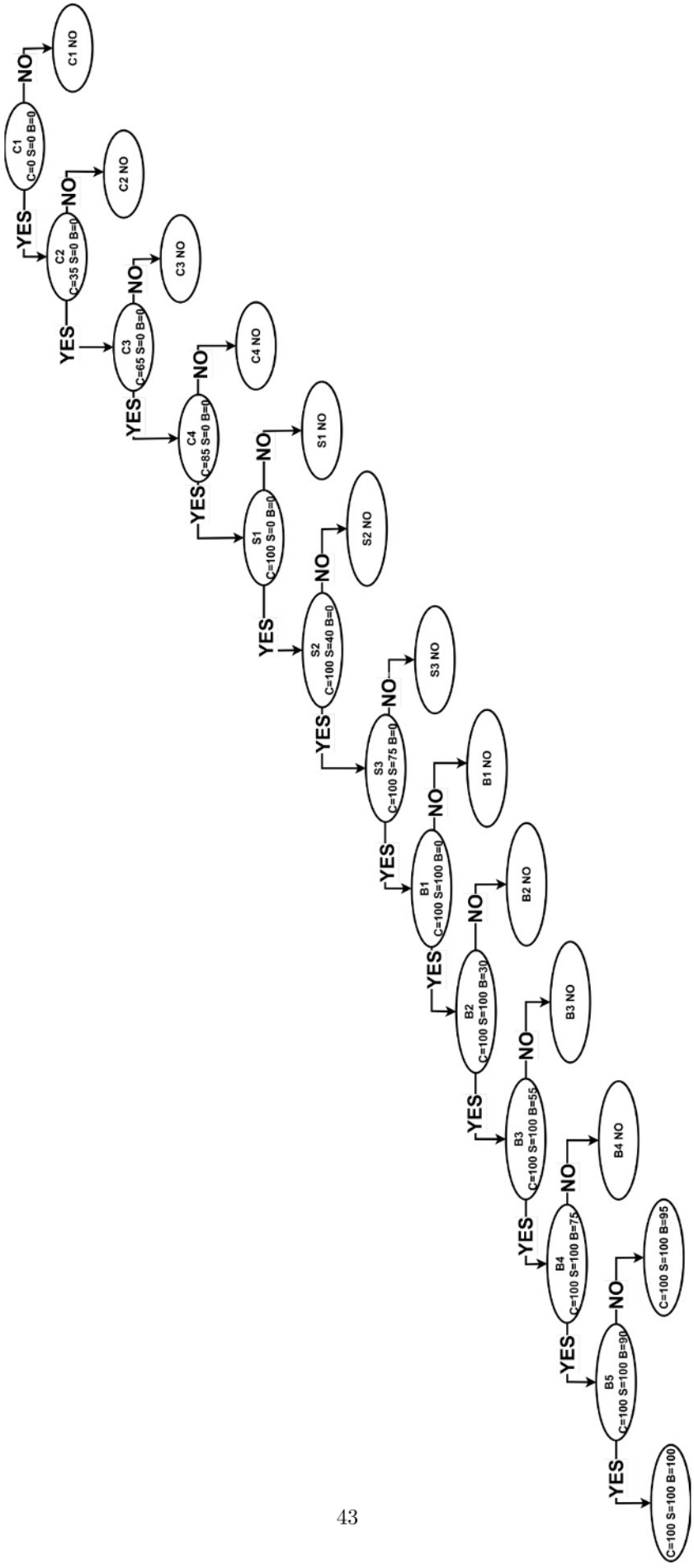


Figure 10: GQM-Based Tree Model

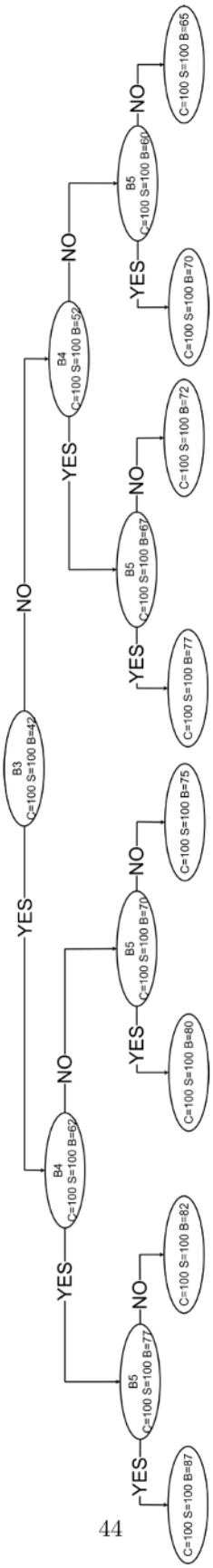


Figure 11: B2 No Subtree

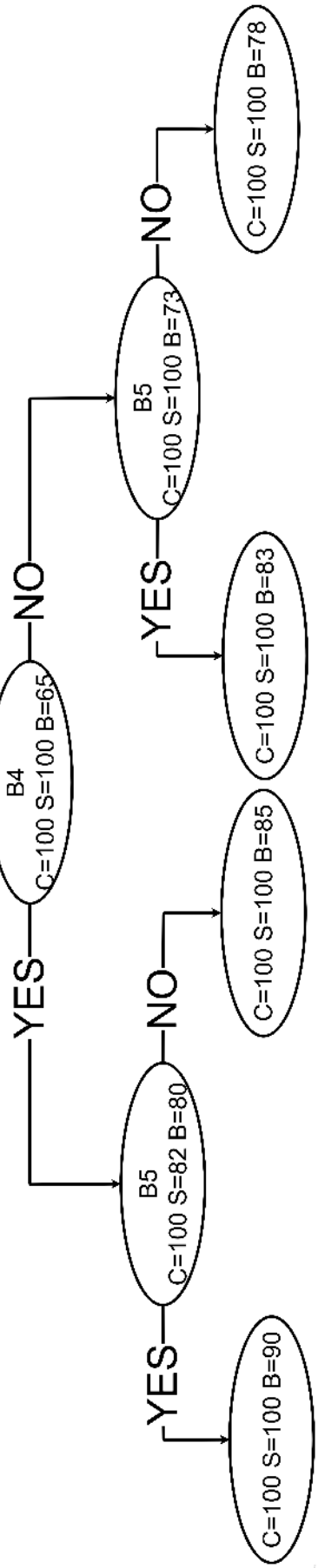


Figure 12: B3 No Subtree

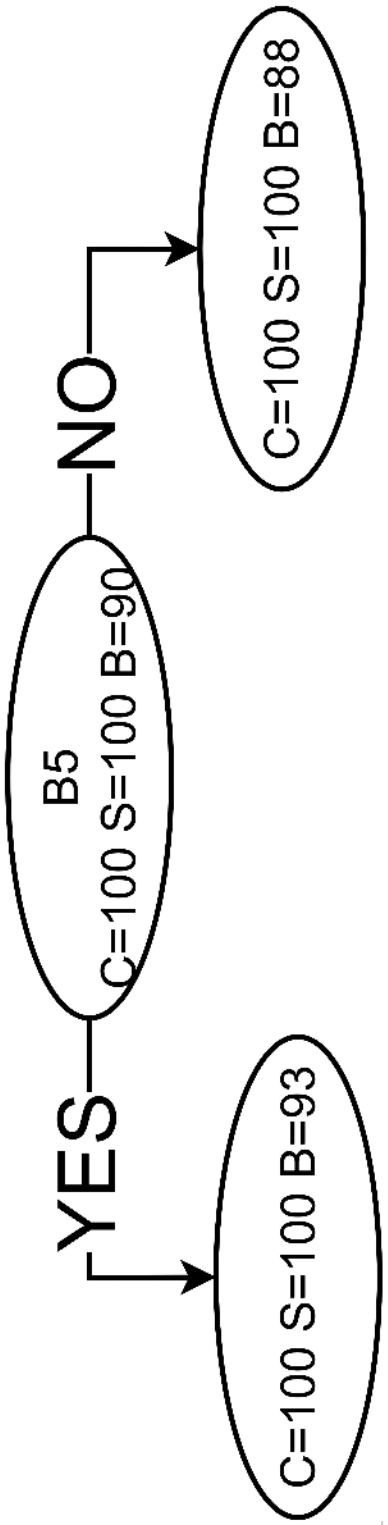


Figure 13: B4 No Subtree

14 Algorithm

In the algorithm below, the steps of how the weights were evaluated will be explained which is based on the Recommendation System for Design Patterns in Software Development Conference Paper **??**.

1. Extract the questions from the definitions and the most common problems that each category solves.
2. Set two answers to each question which are Yes or No only.
3. Set the total weight of the Yes answers regarding each category's questions to 100.
4. Set the total weight of the No answers regarding the categories' questions to 50.
5. Assign the Yes weights of each question depending on the importance of the problem that the question describes, and the No weights which will be the half of its corresponding Yes value. (Example: Yes = 40, No = 20)
6. Calculate the total sum of the categories' weight.
7. Find the highest total weight which will be selected as the suitable category according to the answers given.

15 Experiments

The GQM base tree model was tested on 8 case studies to check if it could recognize the suitable categories (Creational, Structural, Behavioral). These case studies were retrieved from Gang of Four - Design Patterns: Elements of Reusable Object-Oriented Software and Dive into Design Patterns books.

1. Consumers ordering from a catalog. The consumer calls one number and speaks with a customer service representative. The customer service representative providing an interface to the order fulfillment department, the billing department, and the shipping department. Since the sum of the Structural category weight is the maximum, so it'll be chosen as the suitable category to this problem scenario.

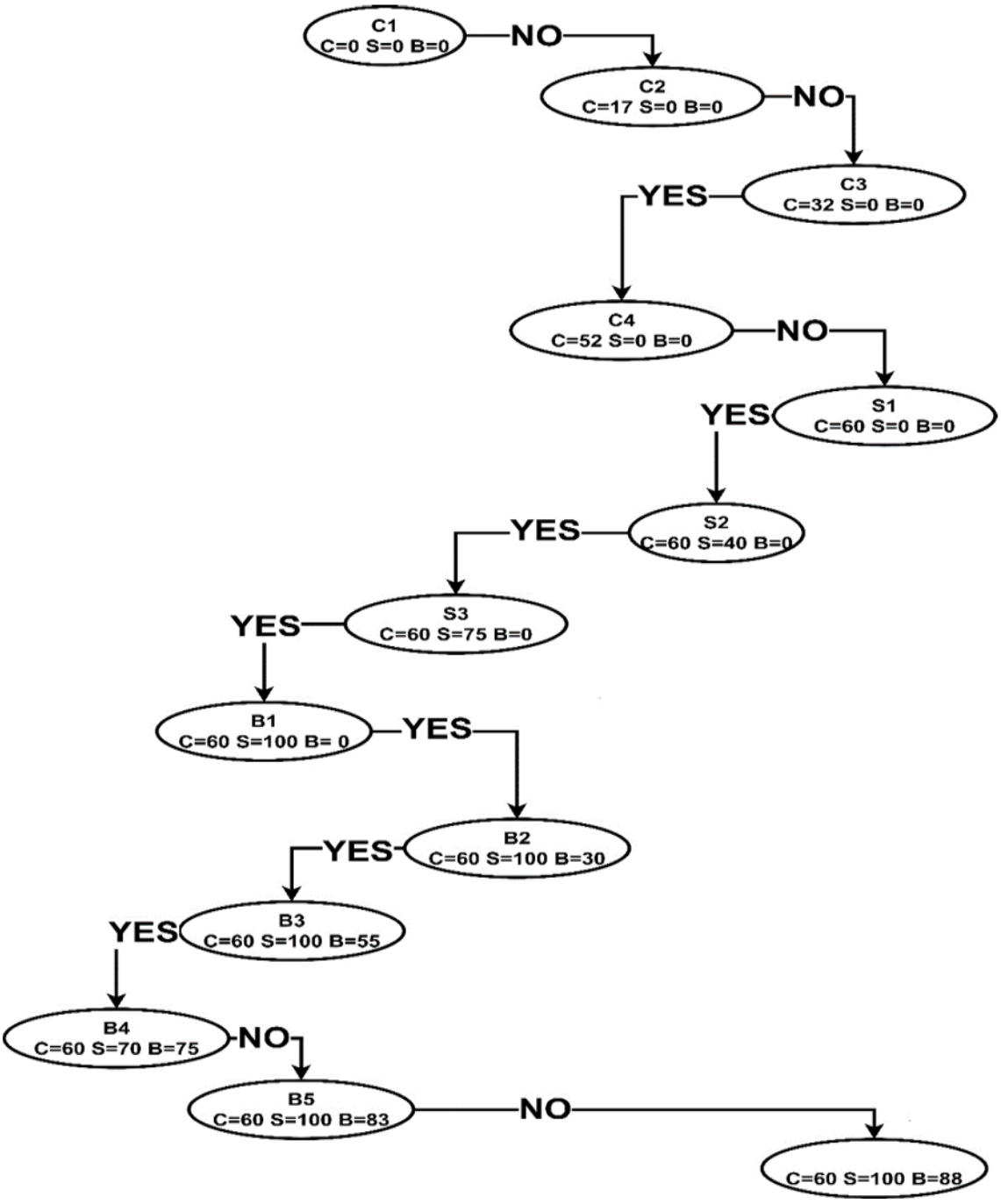


Figure 14: B3 No Subtree

2. The office of the President of the United States is a Singleton. The United States Constitution specifies the means by which a president is elected, limits the term of office, and defines the order of succession. As a result, there can be at most one active president at any given time. Regardless of the personal identity of the active president, the title, "The President of the United States" is a global point of access that identifies the person in the office. Since the sum of the Creational category weight is the maximum, so it'll be chosen as the suitable category to this problem scenario.

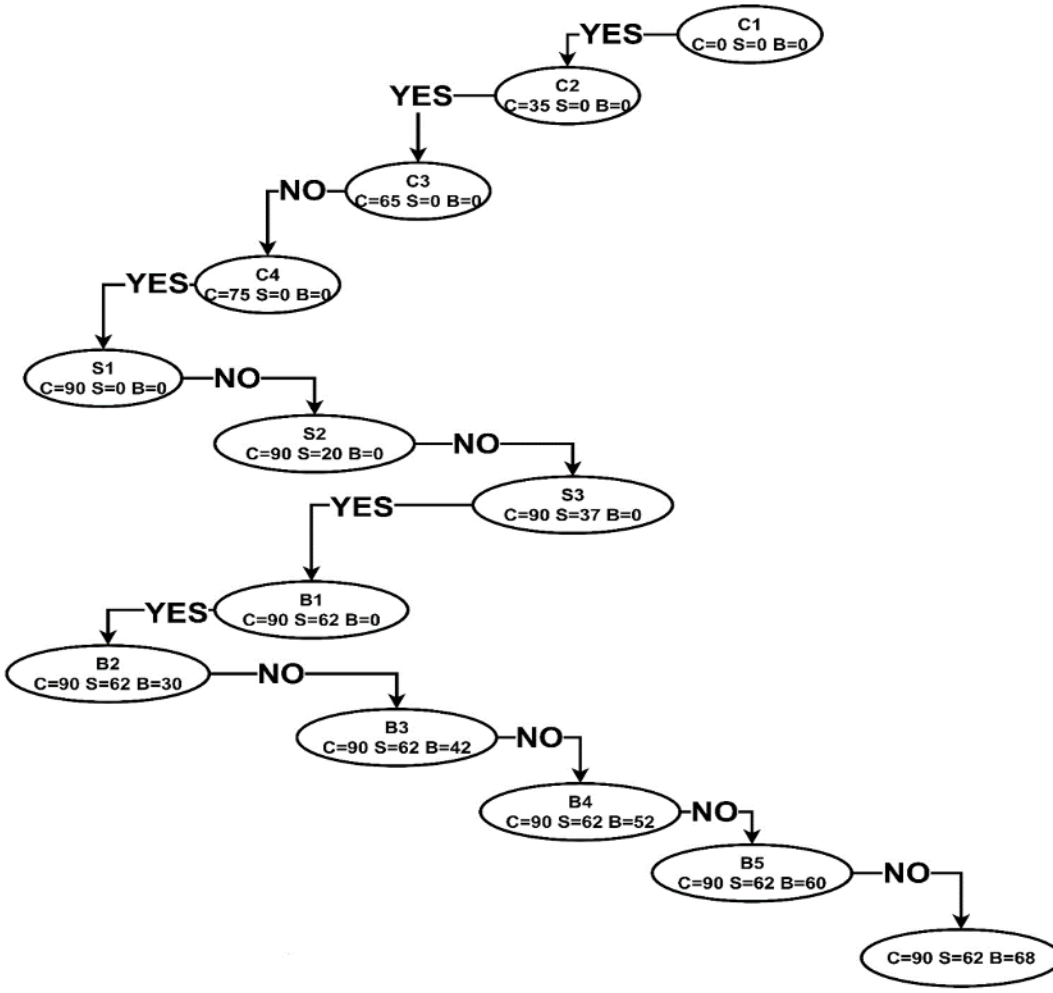


Figure 15: B3 No Subtree

3. Let's say you have a pizza shop, and as a cutting-edge pizza store owner, but you need more than one type of pizza...So then you'd add some code that determines the appropriate type of pizza and then goes about making the pizza. But the pressure is on to add more pizza types. You realize that all of your competitors have added a couple of trendy pizzas to their menus: The Clam Pizza and the Veggie Pizza. Obviously, you need to keep up with the competition, so you'll add these items to your menu. And you haven't been selling many Greek Pizzas lately, so you decide to take that off the menu. Since the sum of the Creational category weight is the maximum, so it'll be chosen as the suitable category to this problem scenario.

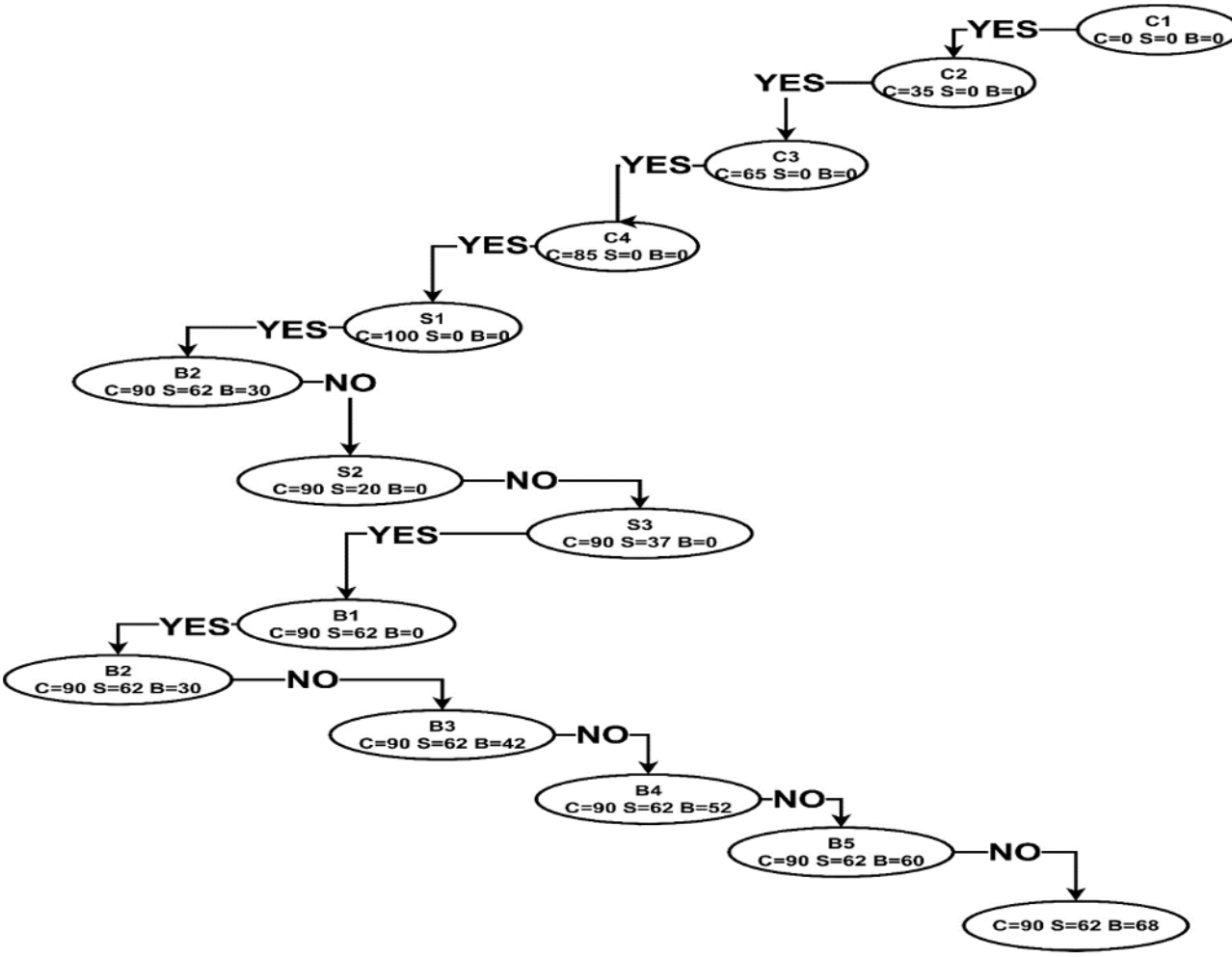


Figure 16: B3 No Subtree

4. A soccer game problem: A user operates a game in the following sequence Start the game Select two teams Add or remove players to/from a team Pick a playground Start a match the system may have a number of PlayGrounds in it, some Teams etc. Player, who plays soccer Team, with various players in it Ball, which is handled by various players. PlayGround, where a match takes place Referee, to control the game Also, you may need some logical objects in your game engine, like Game, which defines a football game, including two teams, a ball, a referee, a playground etc GameEngine to simulate a number of games at a time. TeamStrategy, to decide a team's strategy while playing. Since the sum of the Behavioral category weight is the maximum, so it'll be chosen as the suitable category to this problem scenario.

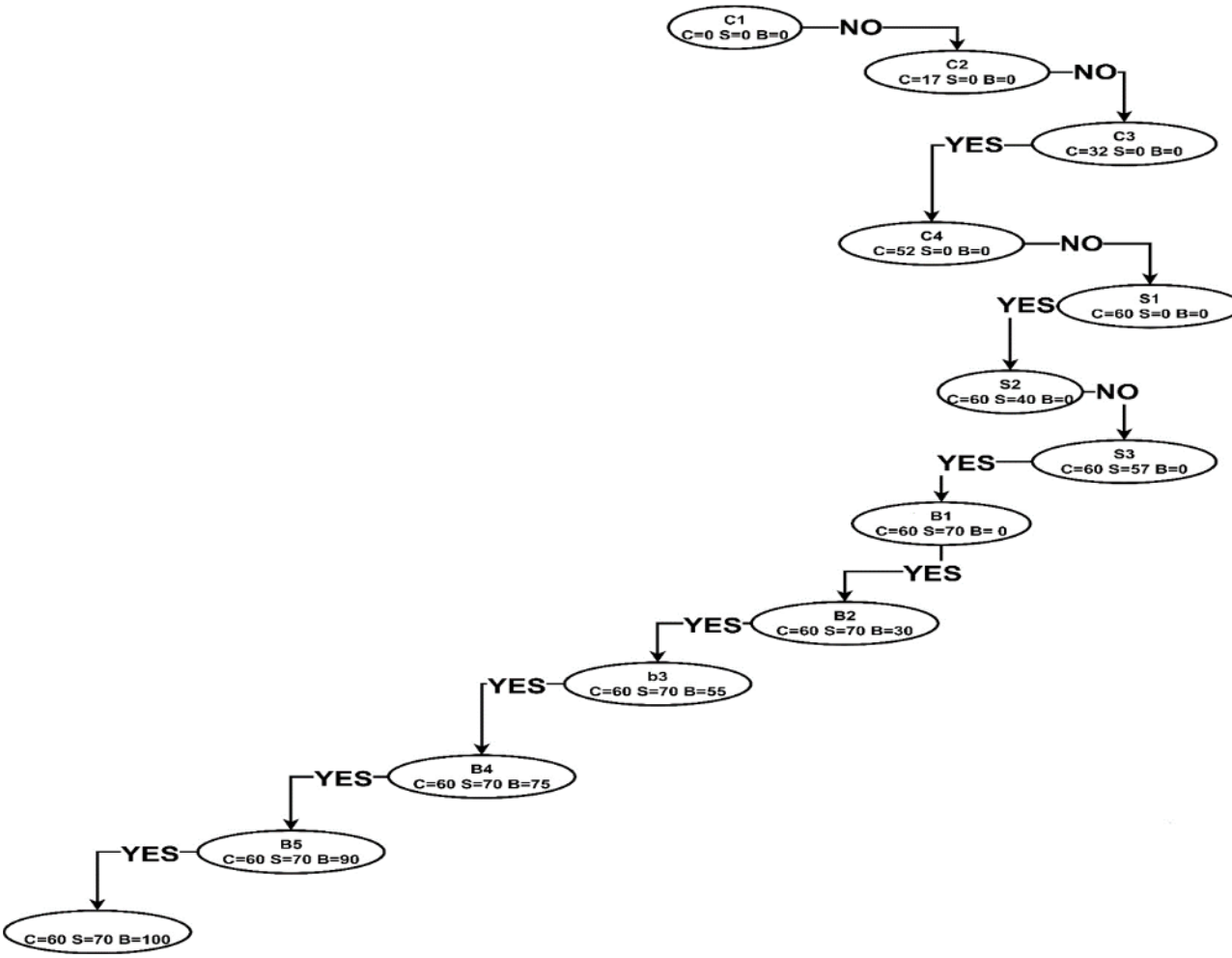


Figure 17: B3 No Subtree

5. Text editor example: Design an editor for documents, documents include text and graphics. GUI with multiple windows. Several operations on texts: spelling, searching, etc. Design alternatives: 1. Different classes for each primitive element: char, line, column, page; and for gliphs like circle, square, etc. 2. Only one abstract class for a generic gliph, with unique interface to implement in different ways. Since the sum of the Structural category weight is the maximum, so it'll be chosen as the suitable category to this problem scenario.

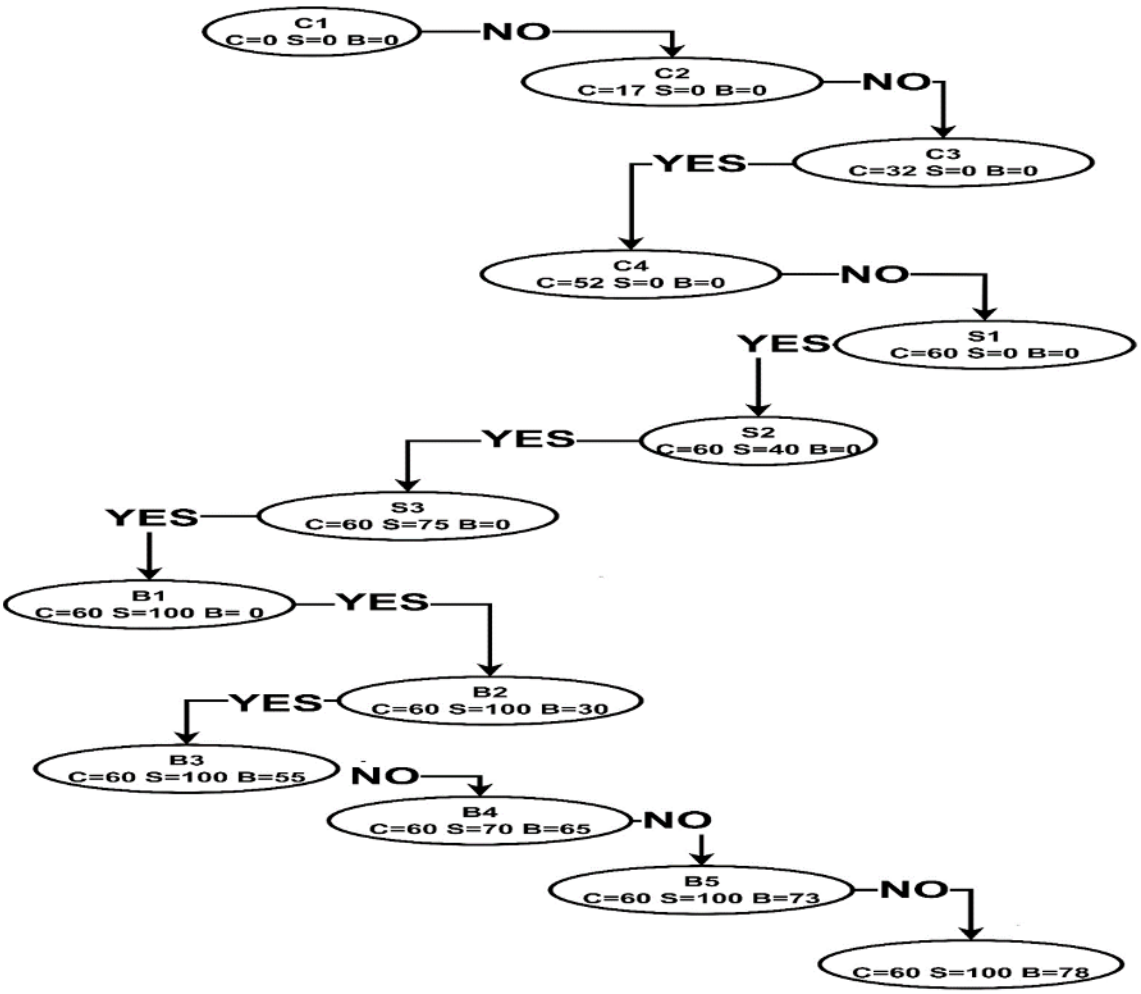


Figure 18: B3 No Subtree

6. The Company class is the central class that encapsulates several important features related to the system as a whole. It is required to make sure that only one instance of this important class can exist. Since the sum of the Creational category weight is the maximum, so it'll be chosen as the suitable category to this problem scenario.

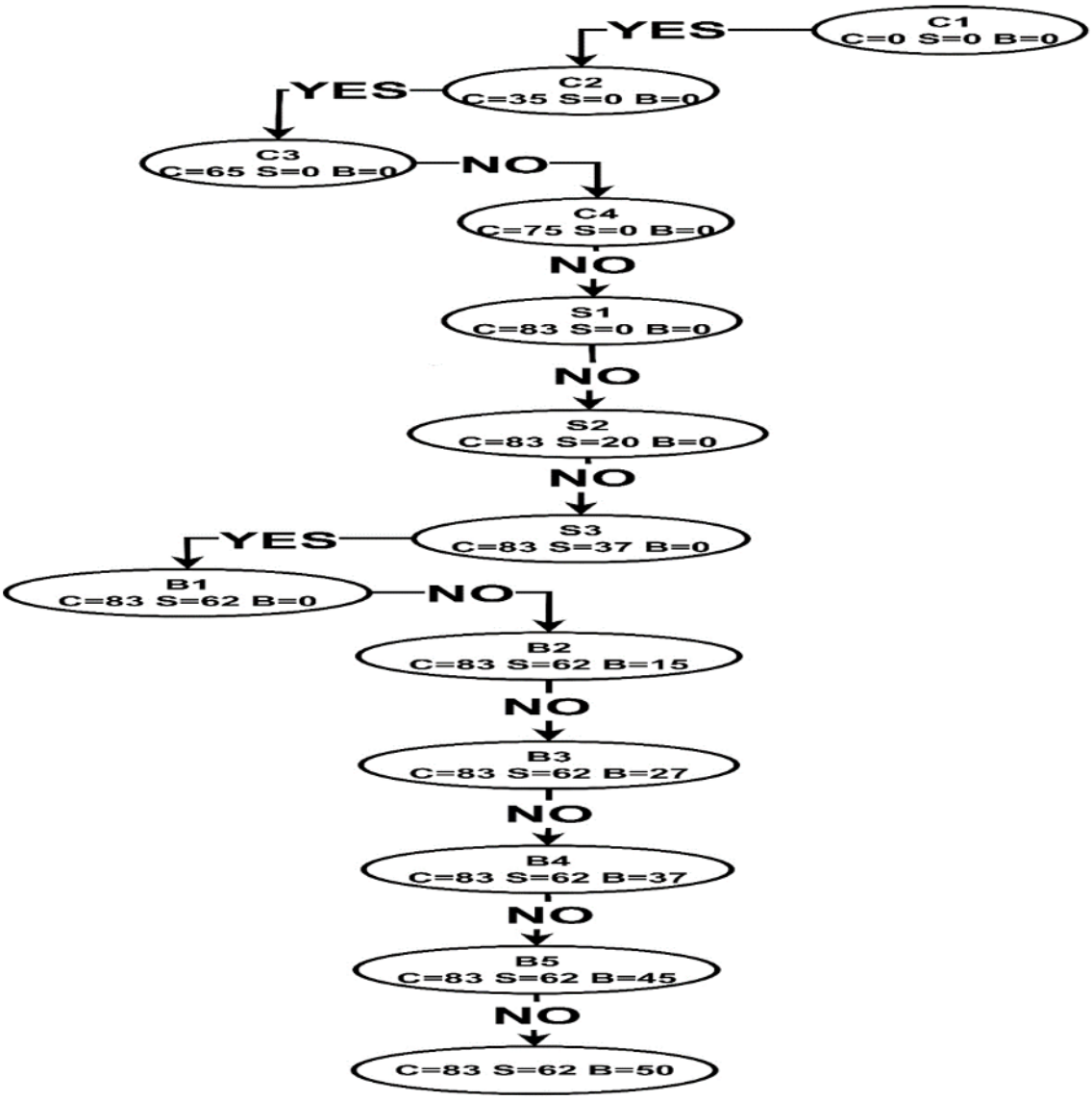


Figure 19: B3 No Subtree

7. Each bidder possesses a numbered paddle that is used to indicate a bid. The auctioneer starts the bidding, and observes when a paddle is raised to accept the bid. The acceptance of the bid changes the bid price which is broadcast to all of the bidders in the form of a new bid. Since the sum of the Behavioral category weight is the maximum, so it'll be chosen as the suitable category to this problem scenario.

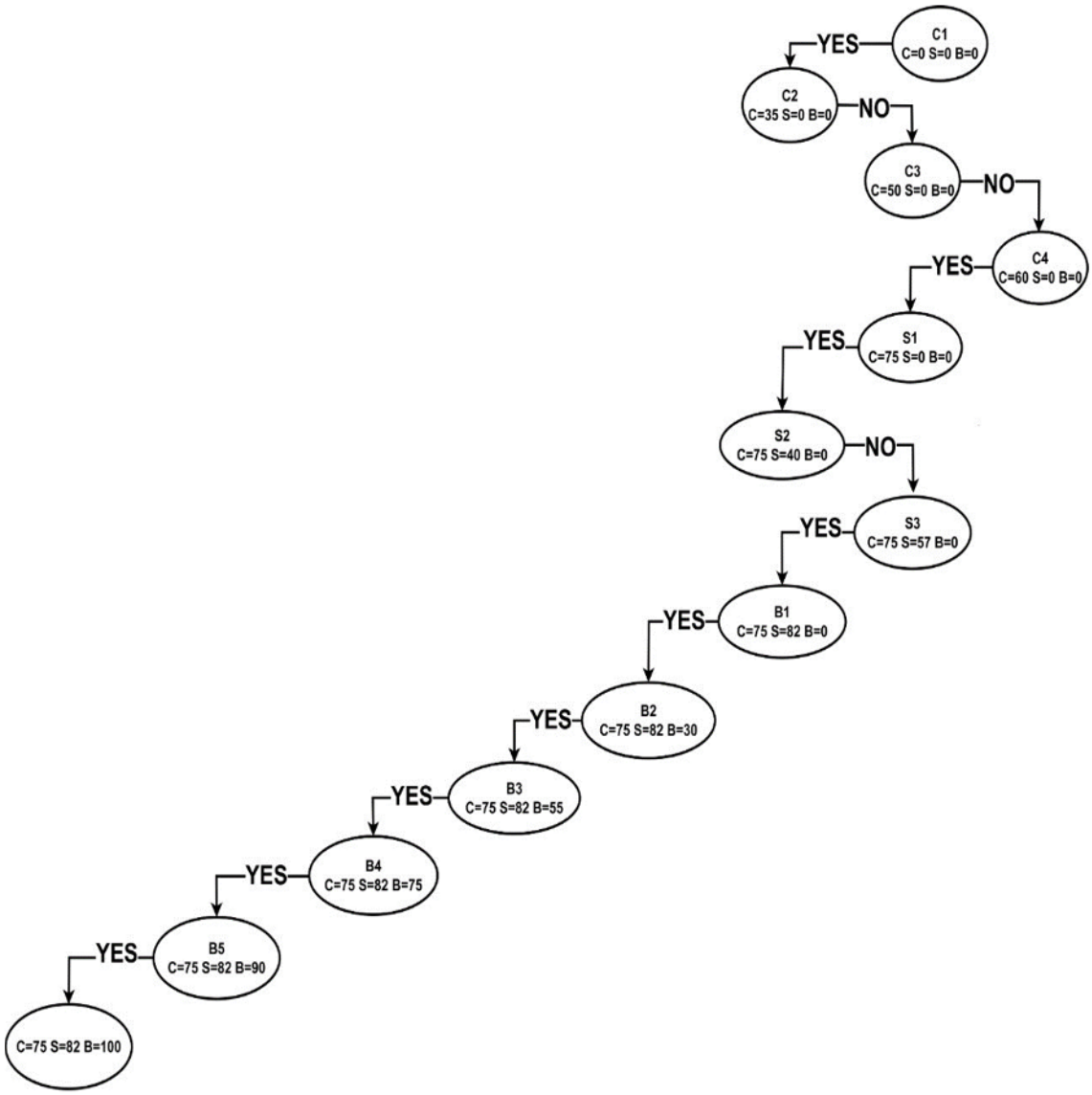


Figure 20: B3 No Subtree

8. A waiter or waitress takes an order or command from a customer and encapsulates that order by writing it on the check. The order is then queued for a short order cook. Note that the pad of "checks" used by each waiter is not dependent on the menu, and therefore they can support commands to cook many different items. Since the sum of the Behavioral category weight is the maximum, so it'll be chosen as the suitable category to this problem scenario.

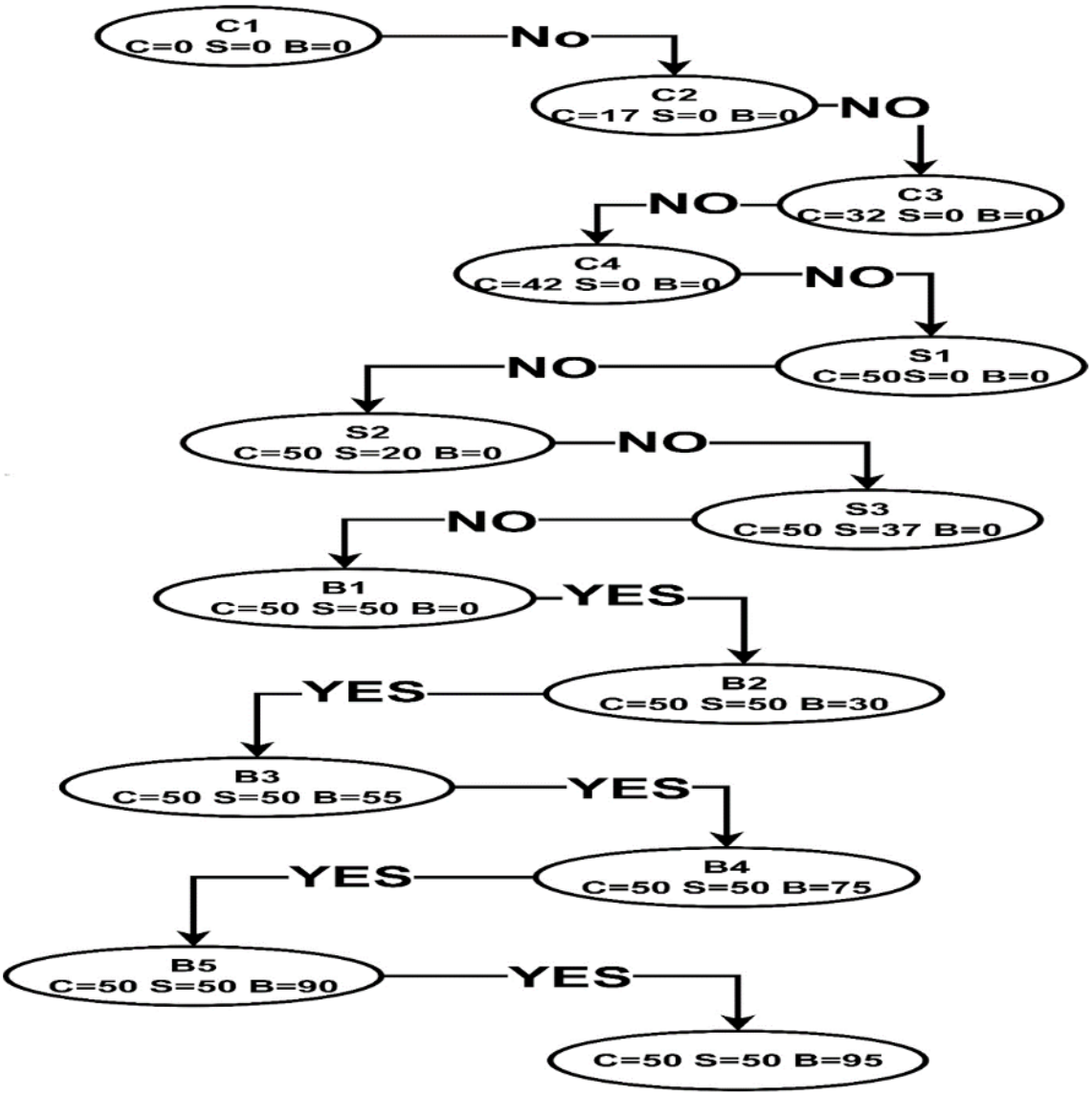


Figure 21: B3 No Subtree

16 References

References

[1] S. Ambler and B. Hanscome, *Process Patterns: Building Large-Scale Systems Using Object Technology*, ser. SIGS: Managing Object Technology. Cambridge University Press, 1998. [Online]. Available: https://books.google.com.eg/books?id=qJJk2yEeoZoC

[2] V. R. Basili, "Software modeling and measurement: the goal/question/metric paradigm," Tech. Rep., 1992.

[3] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of machine Learning research*, vol. 3, no. Jan, pp. 993–1022, 2003.

[4] W. H. Brown, R. C. Malveau, H. W. S. McCormick, and T. J. Mowbray, *AntiPatterns: Refactoring Software, Architectures, and Projects in Crisis*, 1st ed. New York, NY, USA: John Wiley & Sons, Inc., 1998.

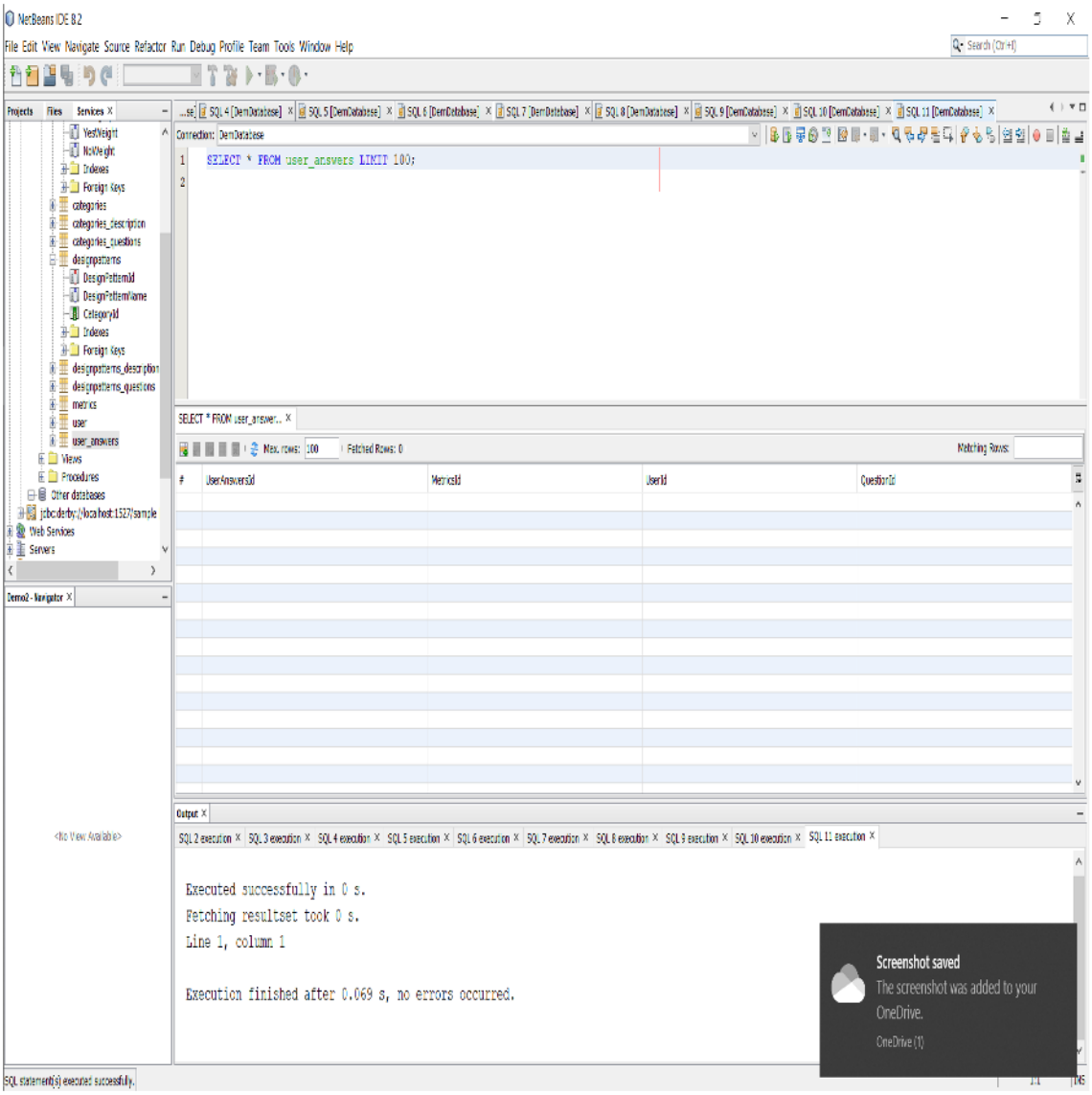
[5] S. Chaturvedi, A. Chaturvedi, A. Tiwari, and S. Agarwal, "Design pattern detection using machine learning techniques," in *2018 7th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions)(ICRITO)*. IEEE, 2018, pp. 1–6.

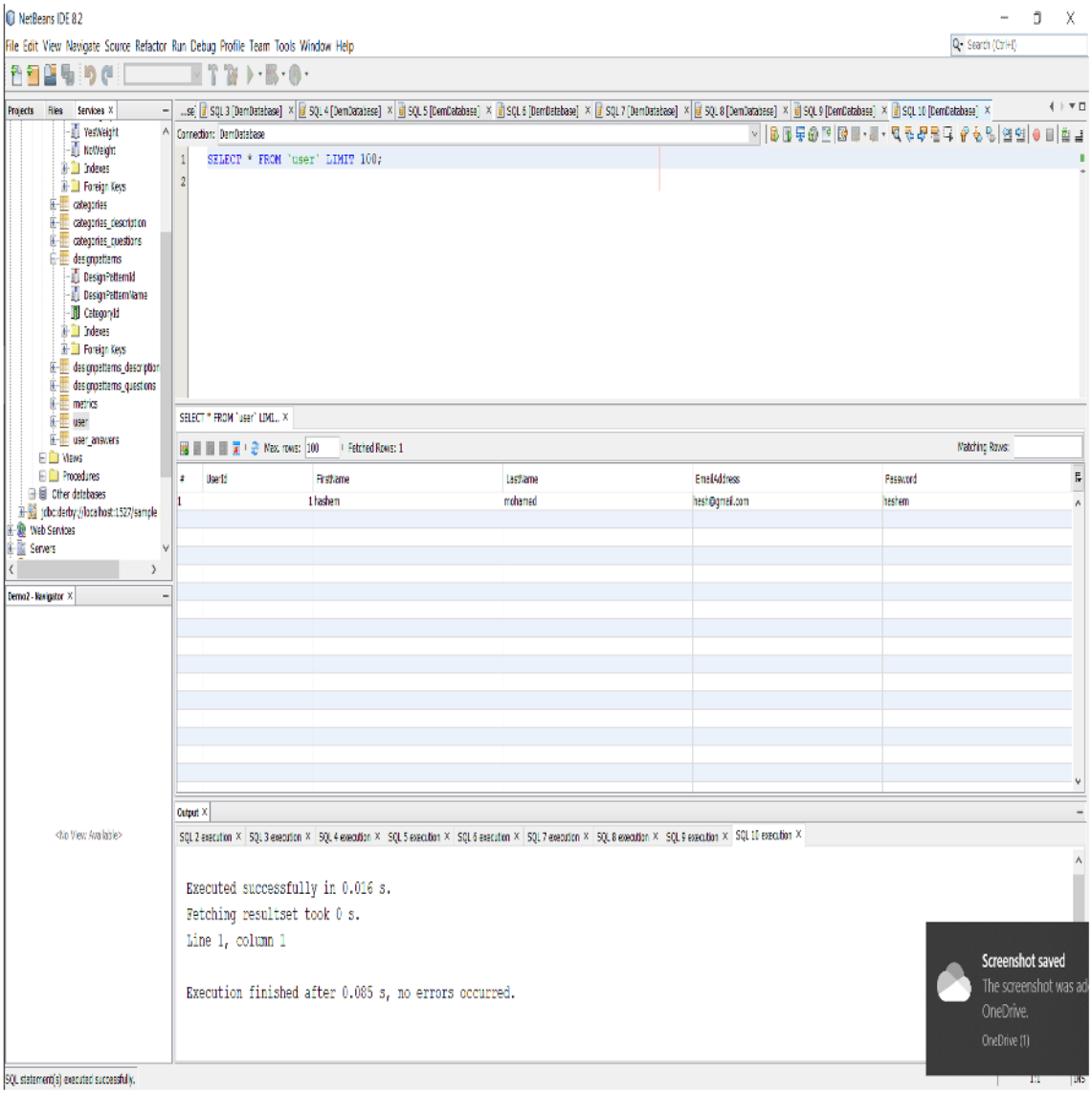
[6] T. Elomaa and H. Koivistoinen, "On autonomous k-means clustering," in *International Symposium on Methodologies for Intelligent Systems*. Springer, 2005, pp. 228–236.

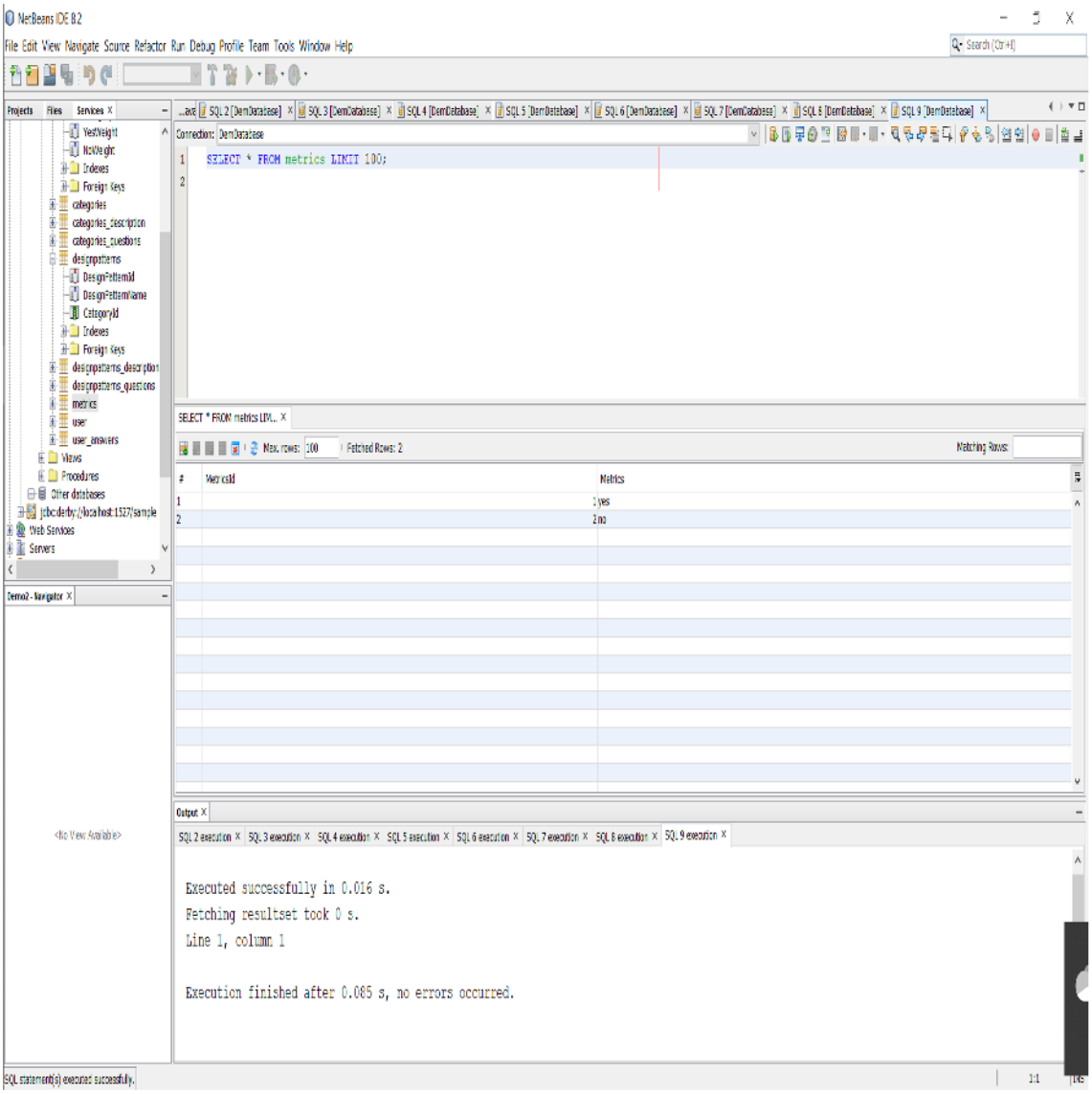
[7] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-oriented Software*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1995.

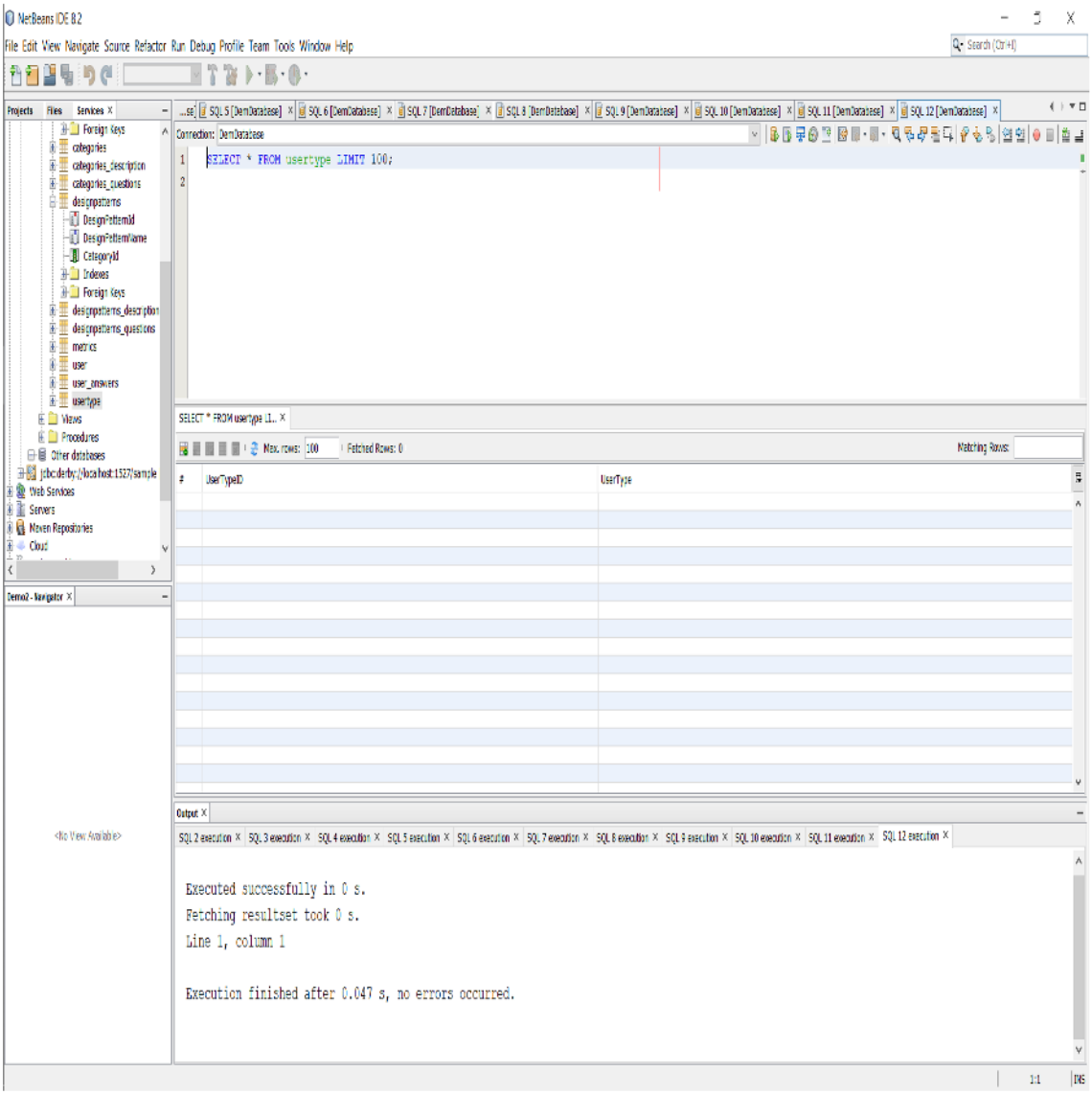
[8] A. Shevts, *Dive Into Design Patterns*. Refactoring.Guru, 2018.

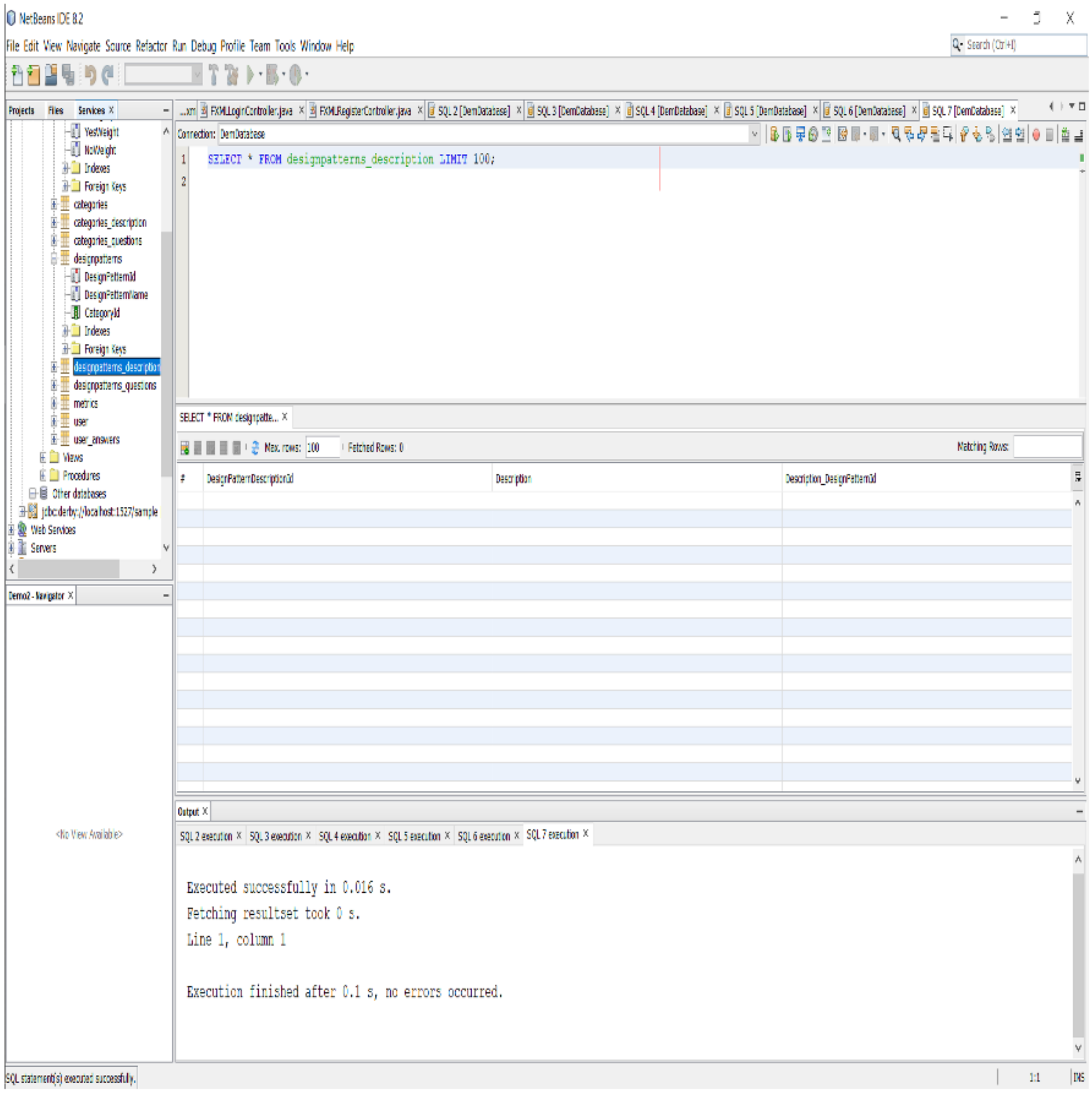
17 Database Schema

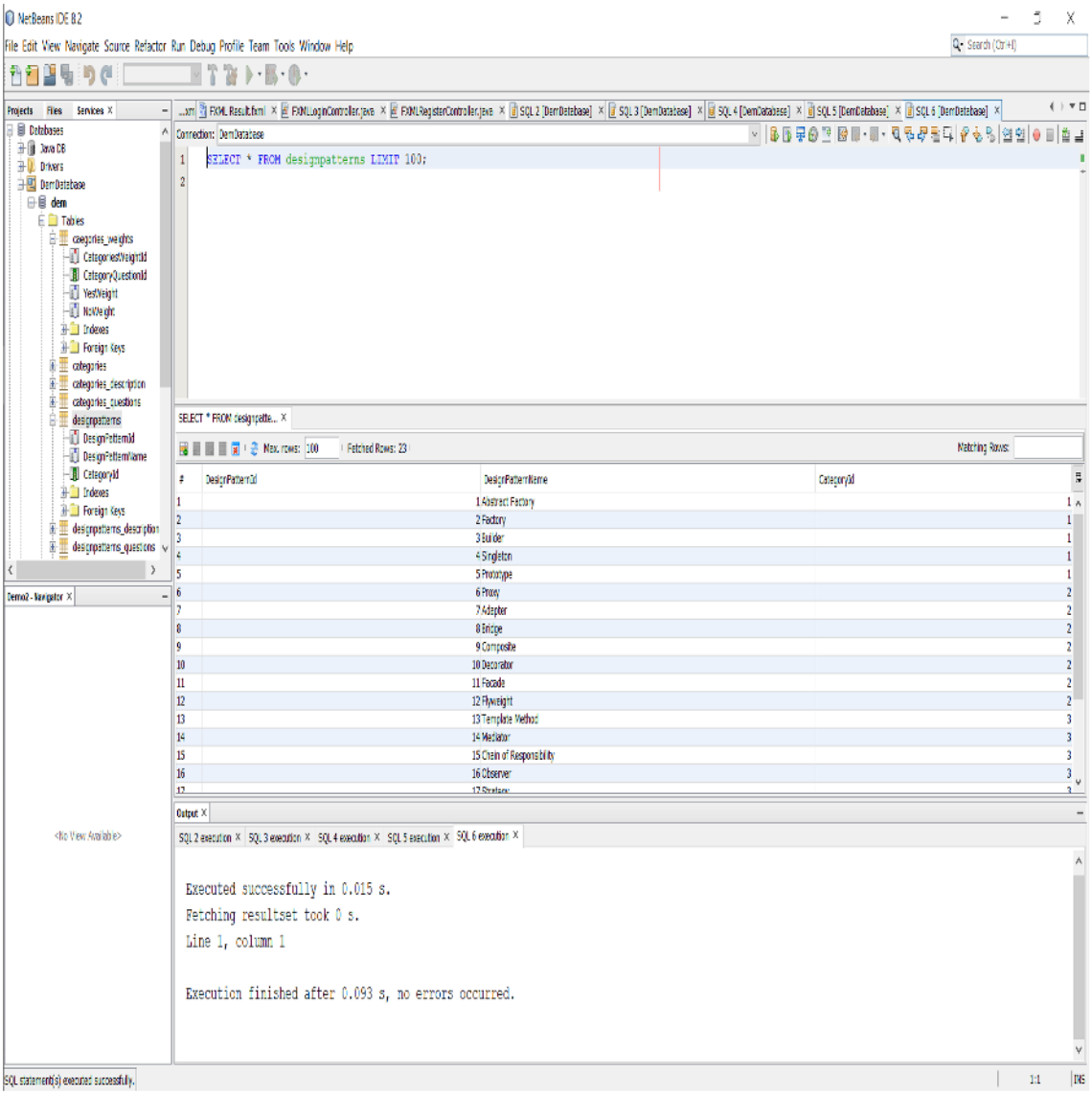


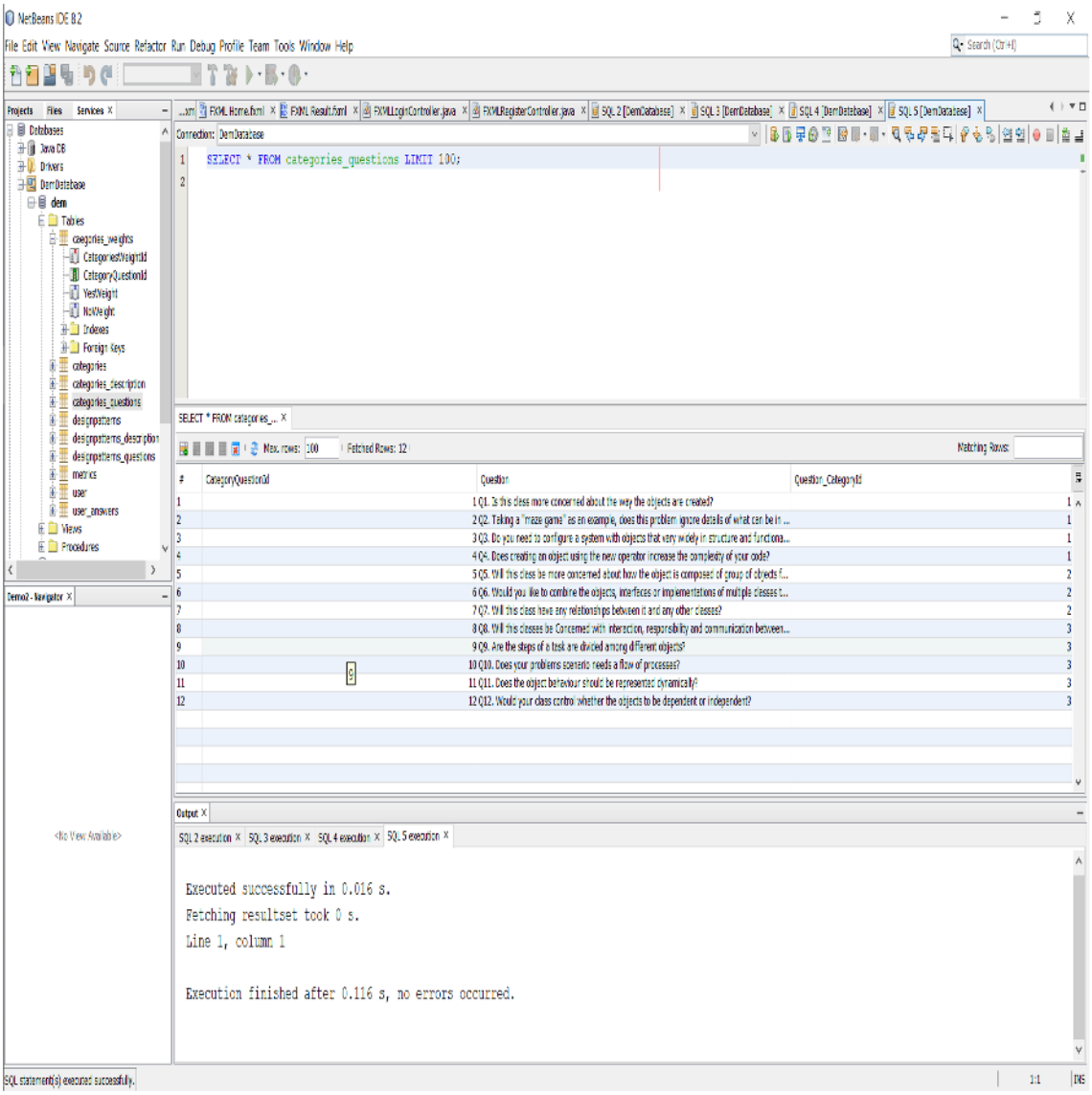

NetBeans IDE 8.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
SELECT * FROM `user` LIMIT 100;
Executed successfully in 0.016 s.
Fetching resultset took 0 s.
Line 1, column 1
Execution finished after 0.085 s, no errors occurred.
Screenshot saved
SQL statement(s) executed successfully.

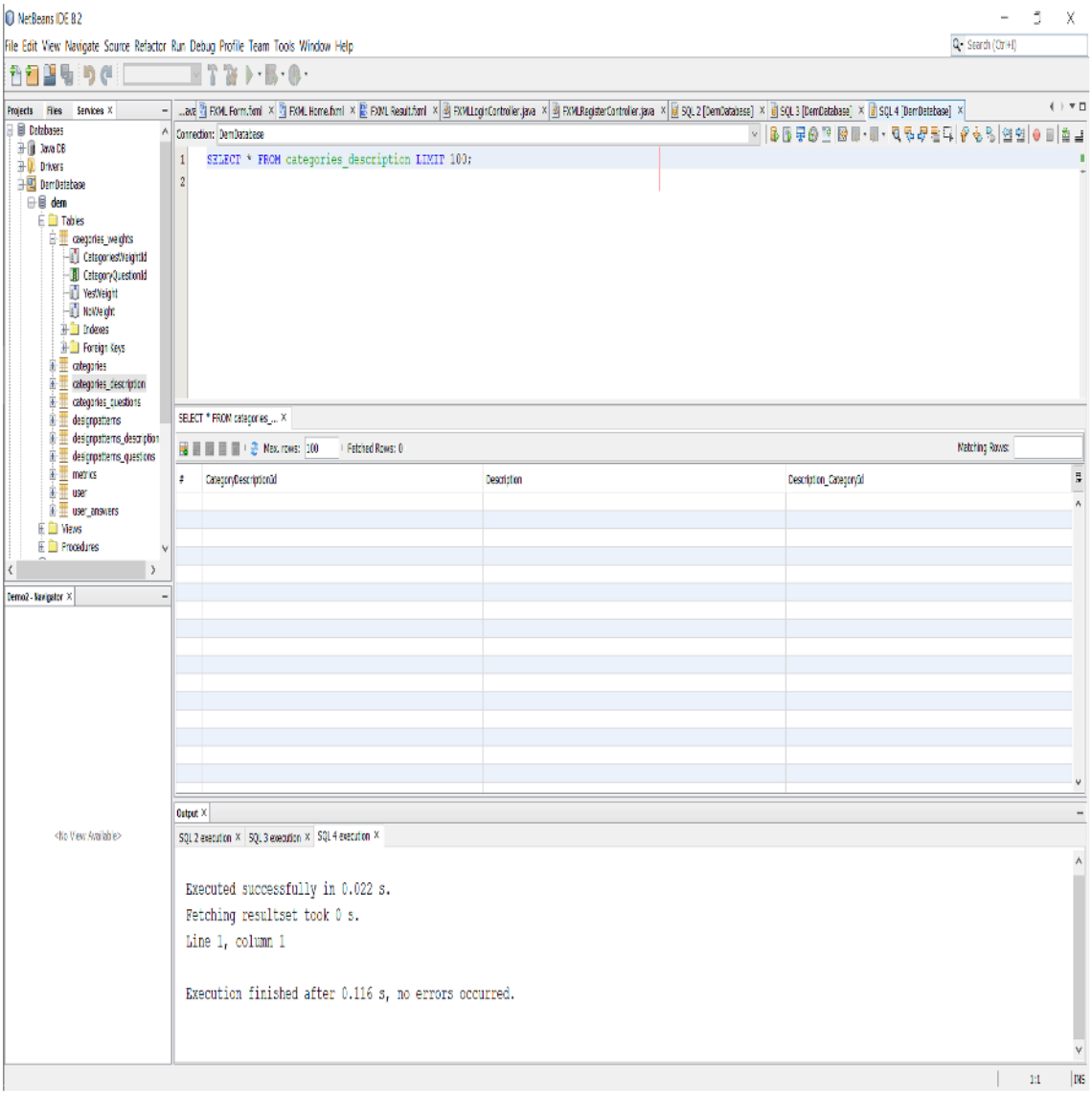

NetBeans IDE 8.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
Projects
Files
Services
Connection: DemDatabase
SELECT * FROM metrics LIMIT 100;
Max. rows: 100
Fetched Rows: 2
Matching Rows:
MetricsId Metrics
1 1 yes
2 2 no
Output
Executed successfully in 0.016 s.
Fetching resultset took 0 s.
Line 1, column 1
Execution finished after 0.085 s, no errors occurred.
SQL statement(s) executed successfully.

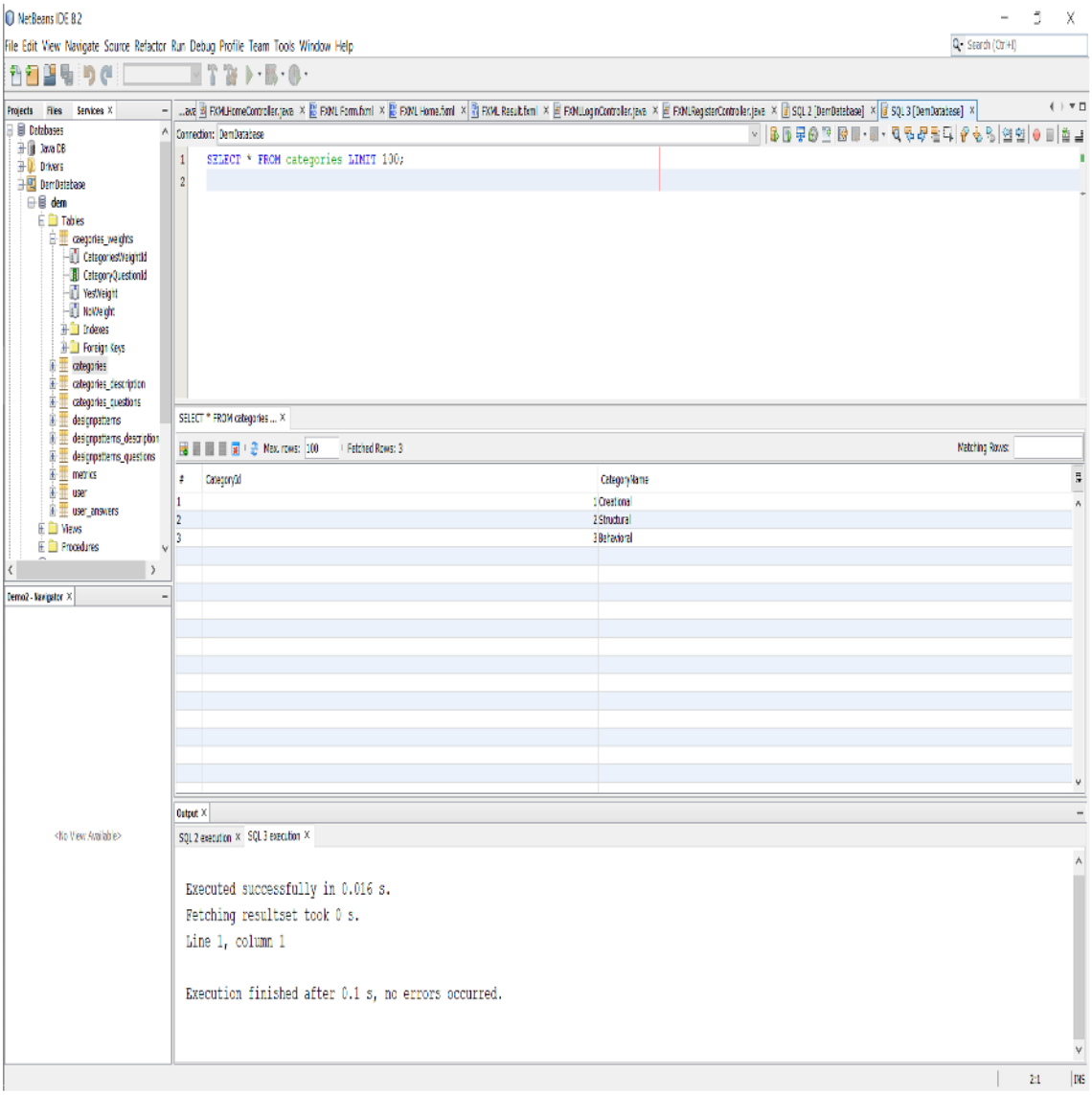

NetBeans IDE 8.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
Projects
Files
Services
Foreign Keys
categories
categories_description
categories_questions
designpatterns
DesignPatternId
DesignPatternName
CategoryId
Indexes
Foreign Keys
designpatterns_description
designpatterns_questions
metrics
user
user_answers
usertype
Views
Procedures
Other databases
jdbc:derby://localhost:1527/sample
Web Services
Servers
Maven Repositories
Cloud
Demo2 - Navigator
<No View Available>
Connection: DemDatabase
1 SELECT * FROM usertype LIMIT 100;
2
SELECT * FROM usertype LI...
Max. rows: 100
Fetched Rows: 0
Matching Rows:
#
UserTypeID
UserType
Output
SQL 2 execution
SQL 3 execution
SQL 4 execution
SQL 5 execution
SQL 6 execution
SQL 7 execution
SQL 8 execution
SQL 9 execution
SQL 10 execution
SQL 11 execution
SQL 12 execution
Executed successfully in 0 s.
Fetching resultset took 0 s.
Line 1, column 1
Execution finished after 0.047 s, no errors occurred.
1:1
INS

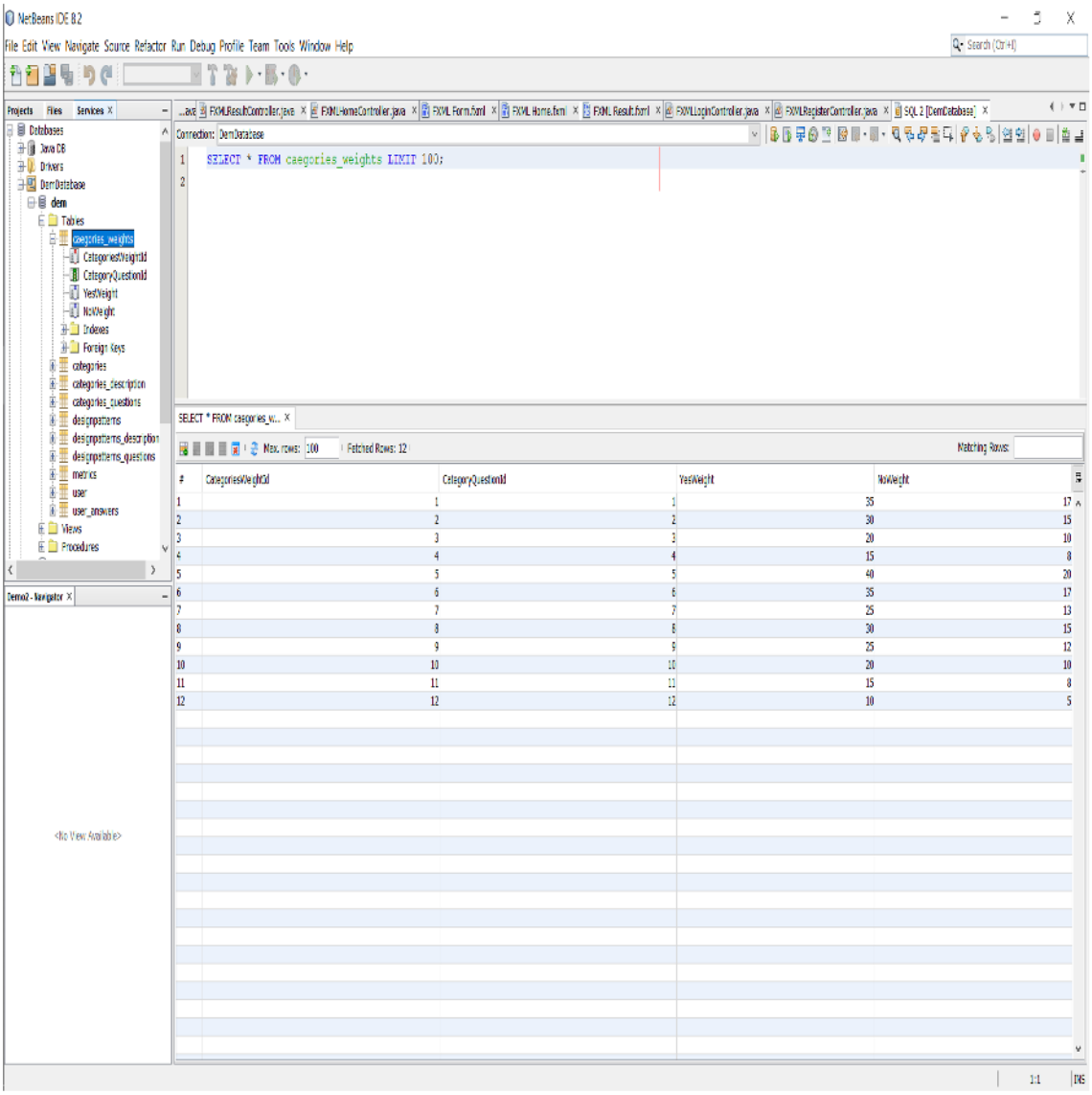

NetBeans IDE 8.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Connection: DemDatabase
SELECT * FROM designpatterns_description LIMIT 100;
SELECT * FROM designpatte...
Max. rows: 100
Fetched Rows: 0
Matching Rows:
DesignPatternDescriptionId
Description
Description_DesignPatternId
Output
Executed successfully in 0.016 s.
Fetching resultset took 0 s.
Line 1, column 1
Execution finished after 0.1 s, no errors occurred.
SQL statement(s) executed successfully.

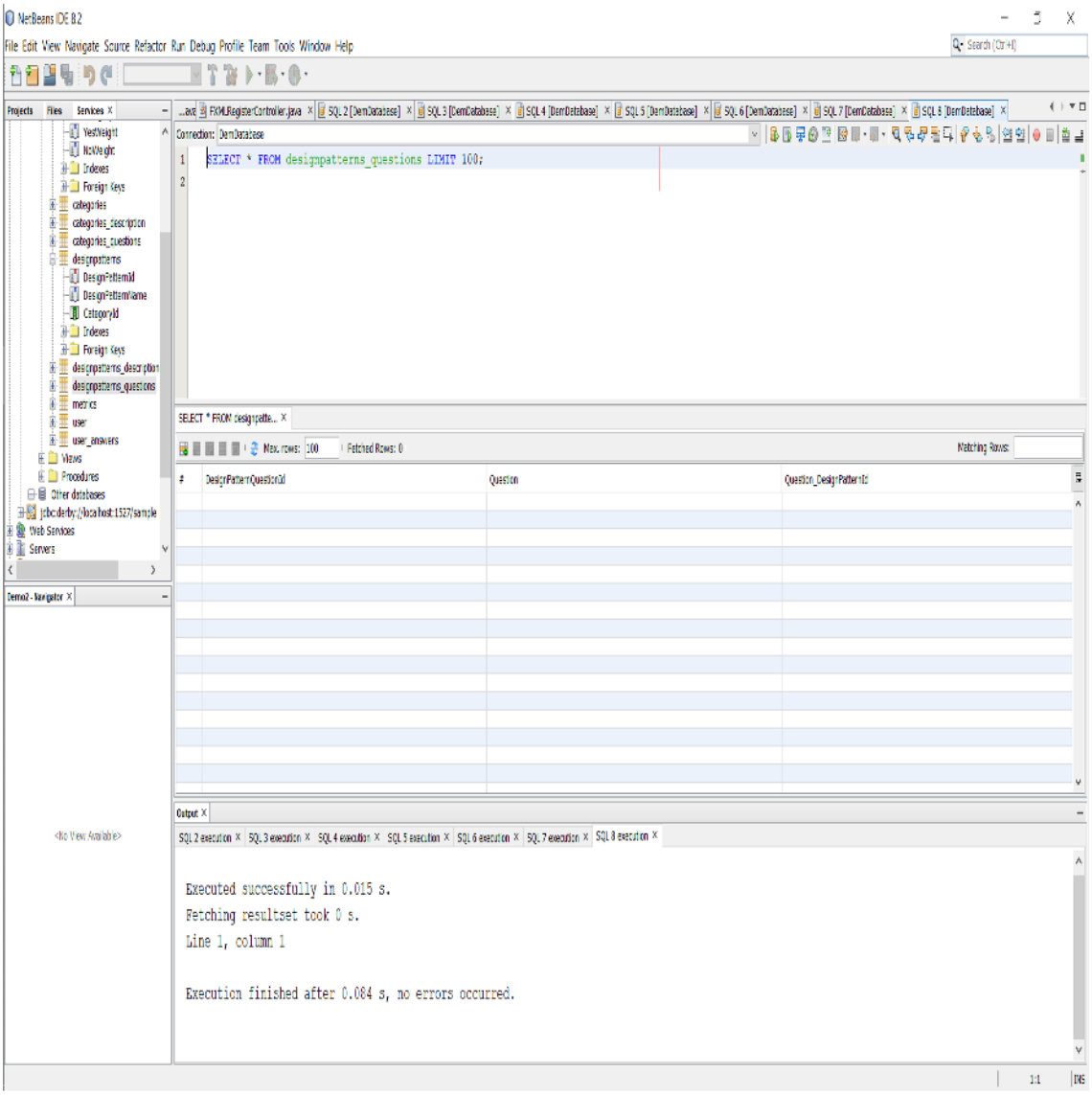

NetBeans IDE 8.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Projects Files Services
Databases
Java DB
Drivers
DemDatabase
dem
Tables
categories_weights
CategoriesWeightId
CategoryQuestionId
YesWeight
NoWeight
Indexes
Foreign Keys
categories
categories_description
categories_questions
designpatterns
DesignPatternId
DesignPatternName
CategoryId
Indexes
Foreign Keys
designpatterns_description
designpatterns_questions
Demo2 - Navigator
<No View Available>
Connection: DemDatabase
1 SELECT * FROM designpatterns LIMIT 100;
2
SELECT * FROM designpatte...
Max. rows: 100
Fetched Rows: 23
Matching Rows:
DesignPatternId DesignPatternName CategoryId
1 1 Abstract Factory 1
2 2 Factory 1
3 3 Builder 1
4 4 Singleton 1
5 5 Prototype 1
6 6 Proxy 2
7 7 Adapter 2
8 8 Bridge 2
9 9 Composite 2
10 10 Decorator 2
11 11 Facade 2
12 12 Flyweight 2
13 13 Template Method 3
14 14 Mediator 3
15 15 Chain of Responsibility 3
16 16 Observer 3
Output
SQL 2 execution SQL 3 execution SQL 4 execution SQL 5 execution SQL 6 execution
Executed successfully in 0.015 s.
Fetching resultset took 0 s.
Line 1, column 1
Execution finished after 0.093 s, no errors occurred.
SQL statement(s) executed successfully.
1:1
INS


NetBeans IDE 8.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
Projects Files Services
Databases
Java DB
Drivers
DemDatabase
dem
Tables
categories_weights
CategoriesWeightId
CategoryQuestionId
YesWeight
NoWeight
Indexes
Foreign Keys
categories
categories_description
categories_questions
designpatterns
designpatterns_description
designpatterns_questions
metrics
user
user_answers
Views
Procedures
Demo2 - Navigator
<No View Available>
Connection: DemDatabase
1 SELECT * FROM categories_questions LIMIT 100;
2
SELECT * FROM categories_...
Max. rows: 100
Fetched Rows: 12
Matching Rows:
CategoryQuestionId Question Question_CategoryId
1 1 Q1. Is this class more concerned about the way the objects are created? 1
2 2 Q2. Taking a "maze game" as an example, does this problem ignore details of what can be in ... 1
3 3 Q3. Do you need to configure a system with objects that vary widely in structure and functiona... 1
4 4 Q4. Does creating an object using the new operator increase the complexity of your code? 1
5 5 Q5. Will this class be more concerned about how the object is composed of group of objects f... 2
6 6 Q6. Would you like to combine the objects, interfaces or implementations of multiple classes t... 2
7 7 Q7. Will this class have any relationships between it and any other classes? 2
8 8 Q8. Will this classes be Concerned with interaction, responsibility and communication between... 3
9 9 Q9. Are the steps of a task are divided among different objects? 3
10 10 Q10. Does your problems scenario needs a flow of processes? 3
11 11 Q11. Does the object behaviour should be represented dynamically? 3
12 12 Q12. Would your class control whether the objects to be dependent or independent? 3
Output
SQL 2 execution SQL 3 execution SQL 4 execution SQL 5 execution
Executed successfully in 0.016 s.
Fetching resultset took 0 s.
Line 1, column 1
Execution finished after 0.116 s, no errors occurred.
SQL statement(s) executed successfully.
1:1
INS


NetBeans IDE 8.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
Projects Files Services
Databases
Java DB
Drivers
DemDatabase
dem
Tables
categories_weights
CategoriesWeightId
CategoryQuestionId
YesWeight
NoWeight
Indexes
Foreign Keys
categories
categories_description
categories_questions
designpatterns
designpatterns_description
designpatterns_questions
metrics
user
user_answers
Views
Procedures
Demo2 - Navigator
<No View Available>
FXML Form.fxml
FXML Home.fxml
FXML Result.fxml
FXMLLoginController.java
FXMLRegisterController.java
SQL 2 [DemDatabase]
SQL 3 [DemDatabase]
SQL 4 [DemDatabase]
Connection: DemDatabase
1 SELECT * FROM categories_description LIMIT 100;
2
SELECT * FROM categories_...
Max. rows: 100
Fetched Rows: 0
Matching Rows:
CategoryDescriptionId Description Description_CategoryId
Output
SQL 2 execution
SQL 3 execution
SQL 4 execution
Executed successfully in 0.022 s.
Fetching resultset took 0 s.
Line 1, column 1
Execution finished after 0.116 s, no errors occurred.
1:1
INS


NetBeans IDE 8.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
Projects Files Services
Databases
Java DB
Drivers
DemDatabase
dem
Tables
categories_weights
CategoriesWeightId
CategoryQuestionId
YesWeight
NoWeight
Indexes
Foreign Keys
categories
categories_description
categories_questions
designpatterns
designpatterns_description
designpatterns_questions
metrics
user
user_answers
Views
Procedures
Demo2 - Navigator
<No View Available>
Connection: DemDatabase
1 SELECT * FROM categories LIMIT 100;
2
SELECT * FROM categories ... X
Max. rows: 100
Fetched Rows: 3
Matching Rows:
CategoryId CategoryName
1 1 Creational
2 2 Structural
3 3 Behavioral
Output X
SQL 2 execution X
SQL 3 execution X
Executed successfully in 0.016 s.
Fetching resultset took 0 s.
Line 1, column 1
Execution finished after 0.1 s, no errors occurred.
2:1
INS


NetBeans IDE 8.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
Projects Files Services
Databases
Java DB
Drivers
DemDatabase
dem
Tables
caegories_weights
CategoriesWeightId
CategoryQuestionId
YesWeight
NoWeight
Indexes
Foreign Keys
categories
categories_description
categories_questions
designpatterns
designpatterns_description
designpatterns_questions
metrics
user
user_answers
Views
Procedures
Demo2 - Navigator
<No View Available>
Connection: DemDatabase
1 SELECT * FROM caegories_weights LIMIT 100;
2
SELECT * FROM caegories_w...
Max. rows: 100
Fetched Rows: 12
Matching Rows:
CategoriesWeightId CategoryQuestionId YesWeight NoWeight
1 1 1 35 17
2 2 2 30 15
3 3 3 20 10
4 4 4 15 8
5 5 5 40 20
6 6 6 35 17
7 7 7 25 13
8 8 8 30 15
9 9 9 25 12
10 10 10 20 10
11 11 11 15 8
12 12 12 10 5
1:1
INS


NetBeans IDE 8.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
Projects
Files
Services
Connection: DemDatabase
SELECT * FROM designpatterns_questions LIMIT 100;
SELECT * FROM designpatte...
Max. rows: 100
Fetched Rows: 0
Matching Rows:
DesignPatternQuestionId
Question
Question_DesignPatternId
Output
SQL 2 execution
SQL 3 execution
SQL 4 execution
SQL 5 execution
SQL 6 execution
SQL 7 execution
SQL 8 execution
Executed successfully in 0.015 s.
Fetching resultset took 0 s.
Line 1, column 1
Execution finished after 0.084 s, no errors occurred.
<No View Available>