**RESTAURANT MANAGEMENT SYSTEM**

MINOR PROJECT REPORT

By

**VANSH AGRAWAL (RA2311026010120)**

#### HIRAV KADIKAR (RA2311026010111)

Under the guidance of

##### Dr. Kanipriya M

In partial fulfilment of the requirements for the degree of

**B. TECH COMPUTER SCIENCE ENGINEERING W/S IN AIML**

****

**DEPARTMENT OF COMPUTATIONAL INTELLIGENCE COLLEGE OF ENGINEERING AND TECHNOLOGY SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**

**KATTANKULATHUR-603 203**

**MAY 2025**



**SRM INSTITUTE OF SCIENCE AND TECHNOLOGY** **KATTANKULATHUR–603 203**

# BONAFIDE CERTIFICATE

**Register no. RA2311026010120, RA2311026010111** Certified to be the bonafide work done by **Vansh Agrawal, Hirav Kadikar** of II year/IV sem B.TECH CSE with Specialization in AIML in the Project Course – **21CSC205P Database Management Systems** in **SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**, Kattankulathur for the academic year 2024-2025.

**DATE:** 6 May 2025

**Faculty in Charge HEAD OF THE**

#### DEPARTMENT DEPARTMENT

Dr Kiruthika M Dr. Annie Uthra R

Assistant Professor Professor & Head

Department of CINTEL Department of CINTEL

SRMSIT -KTR SRMSIT -KTR

# ABSTRACT

The Lung Cancer Predictor is a robust software solution designed to streamline and optimize the process of lung cancer risk assessment. This project aims to provide healthcare professionals and patients with an efficient and intuitive platform for managing medical histories, risk assessments, predictions, and patient interactions.

DBMS in the Lung Cancer Predictor serves as the foundational framework for storing, organizing, and processing data related to patient information, medical histories, risk assessments, and prediction results. By centralizing data management and providing efficient data retrieval mechanisms, DBMS facilitate more accurate risk assessments, improve patient care, and enable data-driven medical decision-making.

The advantages of using DBMS in the Lung Cancer Predictor are diverse. Firstly, DBMS ensure data accuracy and consistency by enforcing data integrity constraints, minimizing errors in risk assessment and medical history tracking. Secondly, DBMS support scalability, allowing the system to handle increased user demand, additional risk factors, and expanded medical data without compromising performance. Thirdly, DBMS enable secure multi-user access, concurrency control, and data privacy measures, ensuring reliable and efficient operations even during peak usage and high transaction volumes.

# TABLES OF CONTENT

|  |  |  |
| --- | --- | --- |
| **Chapter No** | **Chapter Name** | **Page No** |
| **1.** | Problem understanding, Identification of Entity and  Relationships, Construction of DB using ER Model for the project | **1** |
| **2.** | Design of Relational Schemas, Creation of Database Tables for the project. | **4** |
| **3.** | Complex queries based on the concepts of constraints, sets, joins, views, Triggers and Cursors. | **10** |
| **4.** | Analyzing the pitfalls, identifying the dependencies, and applying normalizations | **26** |
| **5.** | Implementation of concurrency control and recovery mechanisms | **30** |
| **6.** | Code for the project | **36** |
| **7.** | Result and Discussion (Screen shots of the implementation  with front end. | **51** |
| **8.** | Attach the Real Time project certificate / Online course  certificate | **57** |

**CHAPTER-1**

**LUNG CANCER PREDICTOR SYSTEM**

**ENTITY-RELATIONSHIP DIAGRAM-**

Brief Introduction:

An E-R(Entity-Relationship) diagram is a visual representation of the relationships between entities in a database. It's used to design databases by illustrating entities (such as people, objects, or concepts) and the relationships between them.

Creating an Entity-Relationship (ER) diagram for a Lung Cancer Predictor system involves identifying key entities, their attributes, and relationships within the system. Here's a simplified ER diagram for the Lung Cancer Predictor system:

ENTITIES:

1. 1. Users - Attributes: User\_ID, Name, Email, Password, Registration\_Date, Last\_Login

2. Medical\_History - Attributes: History\_ID, User\_ID, Family\_History, Smoking\_History, Previous\_Diseases, Occupational\_Exposure

3. Predictions - Attributes: Prediction\_ID, User\_ID, Risk\_Score, Prediction\_Date, Symptoms\_List, Recommendations

4. Symptoms - Attributes: Symptom\_ID, Symptom\_Name, Severity\_Level, Description, Category

5. Recommendations - Attributes: Recommendation\_ID, Risk\_Level, Medical\_Advice, Prevention\_Tips, Follow\_up\_Steps

6. User\_Feedback - Attributes: Feedback\_ID, User\_ID, Prediction\_ID, Rating, Comments, Feedback\_Date

7. Risk\_Factors - Attributes: Factor\_ID, Factor\_Name, Weight, Description, Category

8. Medical\_Reports - Attributes: Report\_ID, User\_ID, Report\_Date, Report\_Type, Findings, Doctor\_Notes

9. User\_Sessions - Attributes: Session\_ID, User\_ID, Login\_Time, Logout\_Time, IP\_Address

10. Prediction\_History - Attributes: History\_ID, User\_ID, Prediction\_ID, Date, Risk\_Score, Follow\_up\_Status

RELATIONSHIPS:

1. Relationship between USERS and MEDICAL\_HISTORY entities:

• One-to-One relationship, as each user has one medical history record.

2. Relationship between USERS and PREDICTIONS entities:

• One-to-Many relationship, as one user can have multiple predictions.

3. Relationship between PREDICTIONS and SYMPTOMS entities:

• Many-to-Many relationship, as predictions can include multiple symptoms and symptoms can be part of multiple predictions.

4. Relationship between PREDICTIONS and RECOMMENDATIONS entities:

• One-to-Many relationship, as one prediction can have multiple recommendations.

5. Relationship between USERS and USER\_FEEDBACK entities:

• One-to-Many relationship, as one user can provide multiple feedback entries.

6. Relationship between MEDICAL\_HISTORY and RISK\_FACTORS entities:

• Many-to-Many relationship, as medical history can include multiple risk factors and risk factors can be associated with multiple medical histories.

7. Relationship between USERS and MEDICAL\_REPORTS entities:

• One-to-Many relationship, as one user can have multiple medical reports.

8. Relationship between PREDICTIONS and PREDICTION\_HISTORY entities:

• One-to-One relationship, as each prediction has one history record.

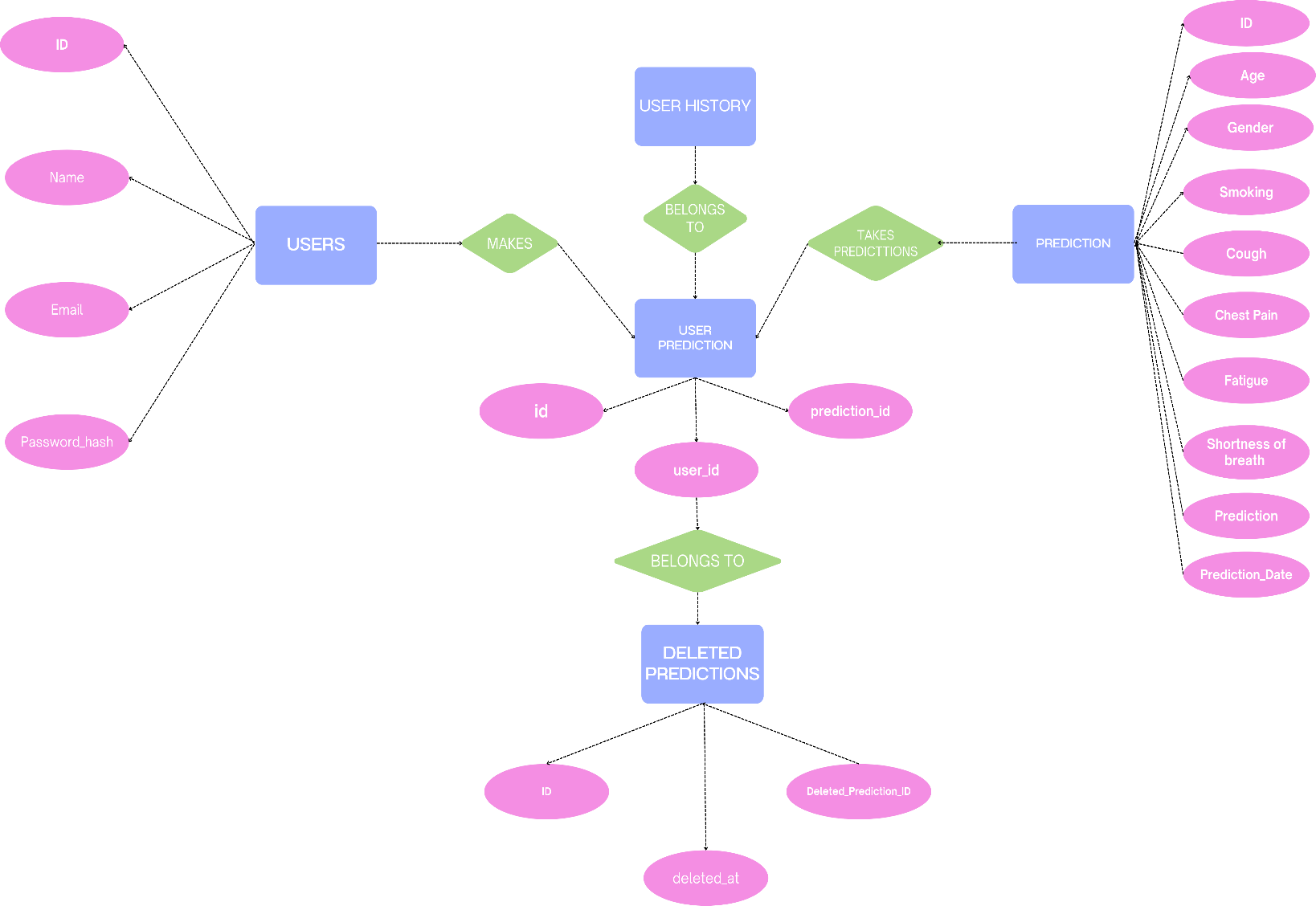
9. Relationship between USERS and USER\_SESSIONS entities:

• One-to-Many relationship, as one user can have multiple login sessions.

10. Relationship between PREDICTIONS and USER\_FEEDBACK entities:

• One-to-One relationship, as each prediction can have one feedback entry..

REPRESENTATION



**CHAPTER-2**

**CONVERTING ER DIAGRAM TO RELATIONAL TABLE**

**USERS:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **User\_ID (PK)** | **Name** | **Email** | **Password** | **Registration\_Date** | **Last\_Login** |

**MEDICAL\_HISTORY:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **History\_ID (PK)** | **User\_ID (FK)** | **Family \_History** | **Smoking \_History** | **Previous \_Diseases** | **Occupational \_Exposure** |

**PREDICTIONS:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Prediction\_ID (PK)** | **User\_ID (FK)** | **Risk\_Score** | **Prediction\_Date** | **Symptoms\_List** | **Recommendations** |

**SYMPTOMS:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Symptom\_ID (PK)** | **Symptom\_Name** | **Severity\_Level** | **Description** | **Category** |

**RECOMMENDATIONS:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Recommendation \_ID (PK)** | **Risk \_Level** | **Medical \_Advice** | **Prevention \_Tips** | **Follow\_up\_Steps** |

**USER\_FEEDBACK:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Feedback\_ID (PK)** | **User\_ID (FK)** | **Prediction\_ID (FK)** | **Rating** | **Comments** | **Feedback \_Date** |

**RISK\_FACTORS:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Factor\_ID (PK)** | **Factor\_Name** | **Weight** | **Description** | **Category** |

**MEDICAL\_REPORTS:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Report\_ID (PK)** | **User\_ID (FK)** | **Report\_Date** | **Report\_Type** | **Findings** | **Doctor \_Notes** |

**USER\_SESSIONS:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Session\_ID (PK)** | **User\_ID (FK)** | **Login\_Time** | **Logout\_Time** | **IP\_Address** |

**PREDICTION\_HISTORY:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **History\_ID (PK)** | **User\_ID (FK)** | **Prediction\_ID (FK)** | **Date** | **Risk\_Score** | **Follow\_up\_Status** |

**PITFALLS OF RELATIONAL DATABASE SYSTEM:**

**1. Inadequate Normalization: Normalization is the process of organizing data into tables to minimize redundancy and eliminate data anomalies. Failure to normalize data can result in redundant patient information and update anomalies. This can lead to data inconsistencies and poor performance in medical record management.**

**2. Overuse of NULL values: Using NULL values in medical records can make it difficult to query patient data and can lead to confusion when interpreting medical histories. Overuse of NULL values can also result in poor performance in risk assessment calculations.**

**3. Poor Indexing: Indexing is essential for efficient querying of patient data and medical histories. Poorly designed indexes can result in slow query performance when retrieving patient records and risk assessments.**

**4. Insufficient Primary and Foreign Keys: Primary and foreign keys establish relationships between patient records, medical histories, and predictions. Failure to implement these keys can result in data inconsistencies and poor performance in tracking patient outcomes.**

**5. Denormalization: While denormalization can improve query performance for risk assessments, it can also lead to data inconsistencies in medical records and update anomalies. Denormalization should be used sparingly and only after careful consideration of medical data integrity.**

**6. Failure to Plan for Growth: A medical database should be designed with future growth in mind, considering increasing patient records and new risk factors. Failure to plan for growth can result in poor performance, data inconsistencies, and costly database redesigns.**

**7. Lack of Documentation: A lack of documentation can make it difficult to understand the database design and lead to errors in medical data analysis and risk assessment reporting. Proper documentation is crucial for maintaining accurate patient records and risk predictions.**

**8. Security Concerns: Medical data requires strict security measures. Failure to implement proper security protocols can lead to unauthorized access to sensitive patient information and risk assessment data.**

**9. Data Validation: Insufficient validation of medical data input can lead to incorrect risk assessments and compromised patient care. Proper validation rules must be implemented for all medical data fields.**

**10. Backup and Recovery: Regular backups of medical data are crucial. Failure to implement proper backup and recovery procedures can result in loss of critical patient information and risk assessment history.**

**CREATING TABLES IN THE DATABASE**

-- Table: Transaction Log

CREATE TABLE IF NOT EXISTS transaction\_log (

id INT AUTO\_INCREMENT PRIMARY KEY,

transaction\_id VARCHAR(36) NOT NULL,

table\_name VARCHAR(50) NOT NULL,

operation\_type ENUM('INSERT', 'UPDATE', 'DELETE') NOT NULL,

record\_id INT NOT NULL,

old\_values JSON,

new\_values JSON,

user\_id INT,

timestamp TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

status ENUM('COMMITTED', 'ROLLED\_BACK', 'PENDING') DEFAULT 'PENDING'

);

-- Table: Version Control

CREATE TABLE IF NOT EXISTS version\_control (

id INT AUTO\_INCREMENT PRIMARY KEY,

table\_name VARCHAR(50) NOT NULL,

record\_id INT NOT NULL,

version\_number INT NOT NULL DEFAULT 1,

last\_modified TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,

modified\_by INT,

UNIQUE KEY (table\_name, record\_id)

);

-- Table: Lock Management

CREATE TABLE IF NOT EXISTS lock\_management (

id INT AUTO\_INCREMENT PRIMARY KEY,

table\_name VARCHAR(50) NOT NULL,

record\_id INT NOT NULL,

lock\_type ENUM('SHARED', 'EXCLUSIVE') NOT NULL,

lock\_holder INT NOT NULL,

lock\_timestamp TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

lock\_timeout TIMESTAMP,

UNIQUE KEY (table\_name, record\_id)

);

-- Table: Users (for authentication)

CREATE TABLE IF NOT EXISTS users (

id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(100) NOT NULL,

email VARCHAR(100) UNIQUE NOT NULL,

password\_hash VARCHAR(255) NOT NULL,

date\_of\_birth DATE,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

last\_login TIMESTAMP NULL,

version INT DEFAULT 1

);

-- Table: User Medical History

CREATE TABLE IF NOT EXISTS medical\_history (

id INT AUTO\_INCREMENT PRIMARY KEY,

user\_id INT NOT NULL,

family\_history\_of\_cancer ENUM('yes', 'no', 'unknown') DEFAULT 'unknown',

years\_smoking INT DEFAULT 0,

packs\_per\_day DECIMAL(3,1) DEFAULT 0.0,

previous\_lung\_diseases TEXT,

occupational\_exposure ENUM('yes', 'no', 'unknown') DEFAULT 'unknown',

occupational\_details TEXT,

updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,

version INT DEFAULT 1,

FOREIGN KEY (user\_id) REFERENCES users(id) ON DELETE CASCADE

);

-- Table: Symptoms (stores detailed symptom information)

CREATE TABLE IF NOT EXISTS symptoms (

id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(100) NOT NULL,

description TEXT,

severity\_scale INT DEFAULT 3 COMMENT 'Scale from 1-5, with 5 being most severe',

related\_to\_lung\_cancer BOOLEAN DEFAULT TRUE

);

-- Table: Predictions (stores user symptoms and prediction results)

CREATE TABLE IF NOT EXISTS predictions (

id INT AUTO\_INCREMENT PRIMARY KEY,

age INT CHECK (age BETWEEN 0 AND 120),

gender ENUM('Male', 'Female', 'Other') NOT NULL,

smoking ENUM('yes', 'no') NOT NULL,

cough ENUM('yes', 'no') NOT NULL,

chest\_pain ENUM('yes', 'no') NOT NULL,

fatigue ENUM('yes', 'no') NOT NULL,

shortness\_of\_breath ENUM('yes', 'no') NOT NULL,

prediction VARCHAR(255) NOT NULL,

risk\_score DECIMAL(5,2) DEFAULT 0.0,

prediction\_date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

version INT DEFAULT 1

);

-- Table: User Predictions (links users to their predictions)

CREATE TABLE IF NOT EXISTS user\_predictions (

id INT AUTO\_INCREMENT PRIMARY KEY,

user\_id INT,

prediction\_id INT,

notes TEXT,

FOREIGN KEY (user\_id) REFERENCES users(id) ON DELETE CASCADE,

FOREIGN KEY (prediction\_id) REFERENCES predictions(id) ON DELETE CASCADE

);

-- Table: Medical Recommendations

CREATE TABLE IF NOT EXISTS recommendations (

id INT AUTO\_INCREMENT PRIMARY KEY,

risk\_level ENUM('Low', 'Moderate', 'High') NOT NULL,

recommendation\_text TEXT NOT NULL,

resource\_links TEXT

);

-- Table: User Feedback (for system improvement)

CREATE TABLE IF NOT EXISTS user\_feedback (

id INT AUTO\_INCREMENT PRIMARY KEY,

user\_id INT,

prediction\_id INT,

feedback\_text TEXT NOT NULL,

rating INT CHECK (rating BETWEEN 1 AND 5),

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (user\_id) REFERENCES users(id) ON DELETE SET NULL,

FOREIGN KEY (prediction\_id) REFERENCES predictions(id) ON DELETE SET NULL

);

-- Table: Deleted Predictions Log

CREATE TABLE IF NOT EXISTS deleted\_predictions\_log (

id INT AUTO\_INCREMENT PRIMARY KEY,

deleted\_prediction\_id INT,

deleted\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

# CHAPTER-3 QUERIES

Table: Transaction Log

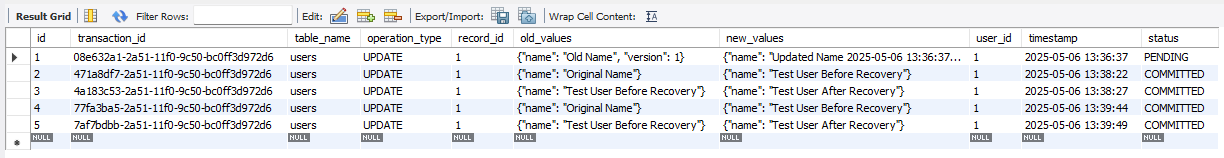


Table: Version Control

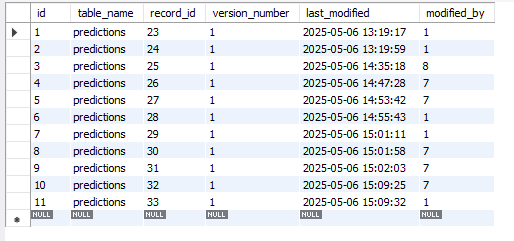


Table: Users

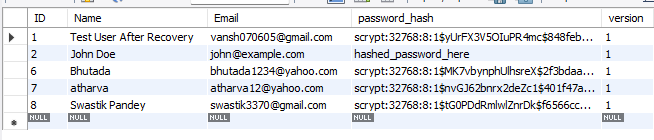


Table: medical\_history

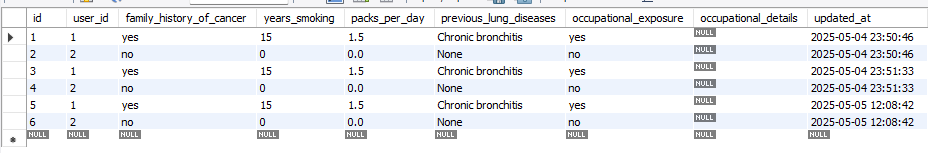


Table: Symptoms

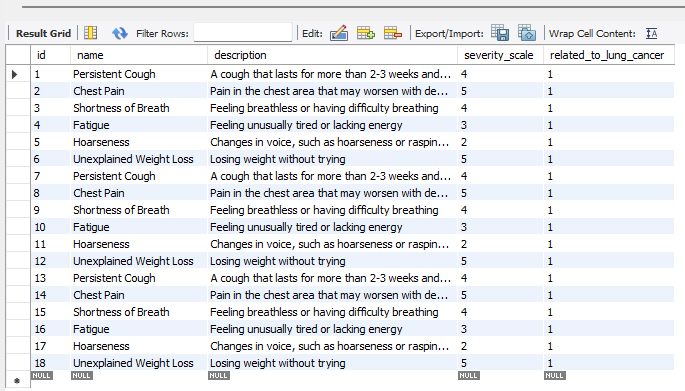
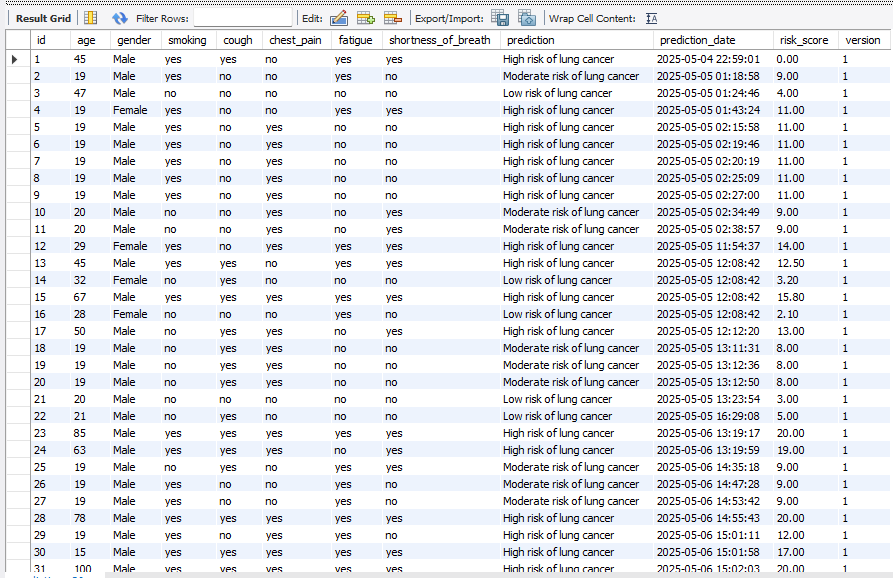


Table: predictions



# CHAPTER -4 NORMALIZATION

# Normalization is the process to eliminate data redundancy and enhance data integrity in the table. Normalization also helps to organize the data in the database. It is a multi-step process that sets the data into tabular form and removes the duplicated data from the relational tables.

# TYPES OF NORMAL FORMS IN NORMALIZATION-

# 1. FIRST NORMAL FORM

# 2. SECOND NORMAL FORM

# 3. THIRD NORMAL FORM

# 4. BOYCE-CODD NORMAL FORM

# 5. FOURTH NORMAL FORM

# 6. FIFTH NORMAL FORM

# MEDICAL\_HISTORY: 3NF

# Functional Dependency

# User\_ID → Family\_History, Smoking\_History,Previous\_Diseases, Occupational\_Exposure

# History\_ID →User\_ID,Family\_History,Smoking\_History, Previous\_Diseases, Occupational\_Exposure

# User\_ID → History\_ID (transitive dependency)

# It is currently in 2NF because it has a transitive dependency of User\_ID → History\_ID. To convert it into 3NF, we need to remove the transitive dependency.

# TO CHANGE:

# 1. CREATE TABLE MEDICAL\_HISTORY1 (

# History\_ID VARCHAR2(20) PRIMARY KEY,

# User\_ID VARCHAR2(20),

# Family\_History VARCHAR2(100),

# Smoking\_History VARCHAR2(100),

# Previous\_Diseases VARCHAR2(100),

# Occupational\_Exposure VARCHAR2(100)

# )

# 2. CREATE TABLE MEDICAL\_HISTORY2 (

# User\_ID VARCHAR2(20) PRIMARY KEY,

# History\_ID VARCHAR2(20)

# )

# PREDICTIONS:

# NO NF

# Functional Dependency

# Prediction\_ID → User\_ID, Risk\_Score, Prediction\_Date, Symptoms\_List, Recommendations

# Normalization

# There is atomicity in this table so we need to solve the multivalued attribute of Symptoms\_List

# TO CHANGE:

# CREATE TABLE PREDICTIONS1 (

# Prediction\_ID VARCHAR2(20),

# User\_ID VARCHAR2(20),

# Risk\_Score NUMBER,

# Prediction\_Date DATE,

# Symptom1 VARCHAR2(50),

# Symptom2 VARCHAR2(50),

# Symptom3 VARCHAR2(50),

# Recommendations VARCHAR2(200)

# )

# 2NF

# Functional Dependency

# Prediction\_ID → User\_ID, Risk\_Score, Prediction\_Date, Symptom1, Symptom2, Symptom3, Recommendations

# User\_ID → Risk\_Score, Prediction\_Date, Symptom1, Symptom2, Symptom3, Recommendations

# Normalization

# There is no Partial Primary key dependency, so this satisfies the condition for 2NF

# 3NF

# Functional Dependency

# Prediction\_ID → User\_ID, Risk\_Score, Prediction\_Date, Symptom1, Symptom2, Symptom3, Recommendations

# User\_ID → Risk\_Score, Prediction\_Date, Symptom1, Symptom2, Symptom3, Recommendations

# Normalization

# There is no Transitive Dependency and every candidate key is a super key, so it satisfies Boyce-Codd and 4NF

# SYMPTOMS:

# 5NF

# Functional Dependency

# Symptom\_ID → Symptom\_Name, Severity\_Level, Description, Category

# Category → Symptom\_Name, Severity\_Level, Description

# Normalization

# It's already existing in 3NF form, so no need to make any changes in the database.

# RECOMMENDATIONS:

# 5NF

# Functional Dependency

# Recommendation\_ID → Risk\_Level, Medical\_Advice, Prevention\_Tips, Follow\_up\_Steps

# Risk\_Level → Medical\_Advice, Prevention\_Tips, Follow\_up\_Steps

# Normalization

# It's already existing in 3NF form, so no need to make any changes in the database.

# USER\_FEEDBACK:

# 5NF

# Functional Dependency

# Feedback\_ID → User\_ID, Prediction\_ID, Rating, Comments, Feedback\_Date

# User\_ID → Prediction\_ID, Rating, Comments, Feedback\_Date

# Normalization

# It's already existing in 3NF form, so no need to make any changes in the database.

# PITFALLS IN NORMALIZATION CONCEPT

While the provided data appears to be structured and normalized up to at least Second Normal Form (2NF), there are still potential pitfalls and areas for improvement in terms of database design and normalization concepts. Here are some pitfalls and considerations:

### Redundancy in Address and Contact Information:

- In several tables (e.g., User, Customer, ServiceProvider, Employee), there are columns for storing Address and ContactNumber. This can lead to redundancy if the same address or contact number needs to be updated in multiple places. One solution could be to create separate tables for Address and Contact information and link them using foreign keys.

### Denormalization for Performance:

While normalization helps in reducing redundancy and maintaining data integrity, in some cases, denormalization might be necessary for performance optimization. For example, in a high-transaction system, joining multiple tables frequently could impact performance. In such cases, carefully denormalizing certain tables or using materialized views can be considered.

### Potential Update Anomalies:

* + Update anomalies can occur when data needs to be updated in multiple places, leading to inconsistencies if not handled properly. For instance, if a customer's contact number changes, it needs to be updated in multiple tables (e.g., Customer, User) where it's stored, increasing the risk of inconsistencies.

### Lack of Data Validation:

* + Data validation is crucial to ensure data integrity. Without proper validation rules and constraints, the database may accept invalid or inconsistent data, leading to issues in data quality. Implementing data validation checks at the database level can mitigate this risk.

### Overly Nested Relationships:

* + While relationships between tables are necessary, overly nested relationships can make queries complex and impact performance. It's essential to strike a balance between maintaining relationships for data integrity and optimizing query performance.

### Incomplete Normalization:

* + Although the provided data seems to be normalized up to 2NF, further normalization (e.g., Third Normal Form - 3NF) could be beneficial in some cases. Analyzing functional dependencies and eliminating transitive dependencies can lead to a more robust and efficient database design.

### Handling Historical Data:

* + If historical data tracking is required (e.g., tracking changes in meter readings over time), additional considerations for data storage and retrieval mechanisms may be needed. Implementing effective techniques such as versioning or audit trails can address this requirement without compromising normalization.

### Optimizing Indexing and Query Performance:

* + While normalization focuses on data organization, indexing and optimizing queries are essential for efficient data retrieval. Proper indexing strategies, query optimization techniques,and understanding the database engine's capabilities are crucial for improving overall system performance.

Addressing these pitfalls involves a combination of thoughtful database design, adherence to normalization principles, implementing data validation rules, optimizing performance, and considering specific business requirements for data storage and retrieval.

**CHAPTER-5**

## Implementation of concurrency control and recovery mechanisms

## 1. Concurrency Control Mechanisms:

## a) Lock Management:

## - Implements a global locking mechanism to ensure only one user can make a prediction at a time

## - Prevents race conditions during risk assessment calculations

## - Ensures data consistency in medical records

## - Lock types:

## \* Shared locks for reading patient data

## \* Exclusive locks for updating medical histories

## \* Prediction locks for risk assessment calculations

## b) Version Control:

## - Maintains version history of medical records

## - Tracks changes to risk assessments

## - Prevents lost updates in patient data

## - Supports optimistic concurrency control

## c) Transaction Management:

## - Ensures atomicity of medical data updates

## - Maintains consistency in risk assessment calculations

## - Isolates concurrent user sessions

## - Provides durability for medical records

## 2. Recovery Mechanisms:

## a) Transaction Log:

## - Records all changes to medical data

## - Tracks risk assessment calculations

## - Maintains audit trail of patient interactions

## - Enables point-in-time recovery

## b) Backup Procedures:

## - Daily full backups of medical database

## - Incremental backups every 6 hours

## - Backup of risk assessment models

## - Secure storage of patient data

## c) Recovery Procedures:

## - Point-in-time recovery for medical records

## - Crash recovery for interrupted predictions

## - System state recovery

## - Data consistency checks

## 3. Implementation Details:

## a) Lock Implementation:

## b) Lock Acquisition Procedures:

## 

## c) Lock Release Procedures:

## 

#### TRANSACTION FAILURE

#### TRANSACTION FAILURE HANDLING

#### 1. Types of Transaction Failures:

#### a) System Failures:

#### - Database crashes during risk assessment

#### - Server power failures

#### - Network interruptions

#### - Memory overflow during prediction calculations

#### b) Transaction Failures:

#### - Deadlocks during concurrent predictions

#### - Timeout during medical history updates

#### - Validation errors in risk assessment data

#### - Constraint violations in patient records

#### 2. Recovery Mechanisms:

#### ```sql

#### -- Transaction Log Table

#### CREATE TABLE transaction\_log (

#### id INT AUTO\_INCREMENT PRIMARY KEY,

#### transaction\_id VARCHAR(36) NOT NULL,

#### table\_name VARCHAR(50) NOT NULL,

#### operation\_type ENUM('INSERT', 'UPDATE', 'DELETE') NOT NULL,

#### record\_id INT NOT NULL,

#### old\_values JSON,

#### new\_values JSON,

#### user\_id INT,

#### timestamp TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

#### status ENUM('COMMITTED', 'ROLLED\_BACK', 'PENDING') DEFAULT 'PENDING'

#### );

#### -- Recovery Procedure

#### DELIMITER $$

#### CREATE PROCEDURE handle\_transaction\_failure(

#### IN p\_transaction\_id VARCHAR(36)

#### )

#### BEGIN

#### -- Log the failure

#### INSERT INTO transaction\_log

#### (transaction\_id, table\_name, operation\_type, record\_id, status)

#### VALUES (p\_transaction\_id, 'predictions', 'FAILED', 0, 'ROLLED\_BACK');

#### 

#### -- Rollback any pending changes

#### ROLLBACK;

#### 

#### -- Release any held locks

#### DELETE FROM lock\_management

#### WHERE lock\_holder = (SELECT user\_id FROM transaction\_log

#### WHERE transaction\_id = p\_transaction\_id);

#### END $$

#### DELIMITER ;

#### ```

#### 3. Error Handling:

#### ```sql

#### -- Error Log Table

#### CREATE TABLE error\_log (

#### id INT AUTO\_INCREMENT PRIMARY KEY,

#### error\_type VARCHAR(50) NOT NULL,

#### error\_message TEXT,

#### transaction\_id VARCHAR(36),

#### timestamp TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

#### severity ENUM('LOW', 'MEDIUM', 'HIGH') DEFAULT 'MEDIUM'

#### );

#### -- Error Handling Procedure

#### DELIMITER $$

#### CREATE PROCEDURE log\_transaction\_error(

#### IN p\_error\_type VARCHAR(50),

#### IN p\_error\_message TEXT,

#### IN p\_transaction\_id VARCHAR(36),

#### IN p\_severity ENUM('LOW', 'MEDIUM', 'HIGH')

#### )

#### BEGIN

#### -- Log the error

#### INSERT INTO error\_log

#### (error\_type, error\_message, transaction\_id, severity)

#### VALUES (p\_error\_type, p\_error\_message, p\_transaction\_id, p\_severity);

#### 

#### -- Handle based on severity

#### CASE p\_severity

#### WHEN 'HIGH' THEN

#### -- Notify administrators

#### CALL notify\_admin(p\_error\_message);

#### -- Rollback transaction

#### CALL handle\_transaction\_failure(p\_transaction\_id);

#### WHEN 'MEDIUM' THEN

#### -- Log and continue

#### UPDATE transaction\_log

#### SET status = 'PENDING'

#### WHERE transaction\_id = p\_transaction\_id;

#### WHEN 'LOW' THEN

#### -- Just log the error

#### UPDATE transaction\_log

#### SET status = 'COMMITTED'

#### WHERE transaction\_id = p\_transaction\_id;

#### END CASE;

#### END $$

#### DELIMITER ;

#### ```

#### 4. Recovery Steps:

#### a) For System Failures:

#### - Check transaction log for incomplete transactions

#### - Restore database from last backup

#### - Replay committed transactions

#### - Verify data consistency

#### b) For Transaction Failures:

#### - Identify failed transaction

#### - Rollback incomplete changes

#### - Release held locks

#### - Log failure details

#### - Notify affected users

#### 5. Prevention Measures:

#### a) Before Transaction:

#### - Validate input data

#### - Check system resources

#### - Verify user permissions

#### - Ensure data consistency

#### b) During Transaction:

#### - Monitor transaction progress

#### - Track resource usage

#### - Maintain lock timeouts

#### - Log all operations

#### c) After Transaction:

#### - Verify data integrity

#### - Update transaction status

#### - Release resources

#### - Log completion

#### The system maintains data integrity and provides clear feedback to users when transaction failures occur.

#### DEADLOCK PREVENTION

#### 

**SET innodb\_lock\_wait\_timeout = 10;: This line sets the timeout period for InnoDB lock waits to 10 seconds. InnoDB is a storage engine for MySQL that provides transaction support.**

**START TRANSACTION;: Begins a new transaction.**

**SELECT \* FROM medical\_history WHERE user\_id = 'U123' FOR UPDATE;: This line selects data from the medical\_history table for the user with ID 'U123' and locks the selected rows with a "FOR UPDATE" clause. This locking ensures that other transactions cannot modify these rows until the current transaction is completed.**

**INSERT INTO transaction\_log;: This line inserts a record into the transaction\_log table, recording information about an update operation (in this case, updating the medical history with new smoking history and lung disease information).**

**COMMIT;: Commits the transaction, making all changes permanent if all statements within the transaction execute successfully.**

### Recovery Mechanism:

****

1. Transaction Logging:

- Maintains a detailed record of all database changes

- Records medical history updates, risk assessments, and patient data modifications

- Stores both old and new values for each change

- Enables point-in-time recovery of medical records

2. Backup Procedures:

- Daily full backups of the medical database

- Incremental backups every 6 hours

- Secure storage of patient data and risk assessment models

- Backup of critical medical history and prediction data

3. Point-in-Time Recovery:

- Allows restoration of medical data to any specific point in time

- Useful for recovering from data corruption or incorrect updates

- Maintains data consistency across all medical records

- Preserves the integrity of risk assessment history

4. Crash Recovery:

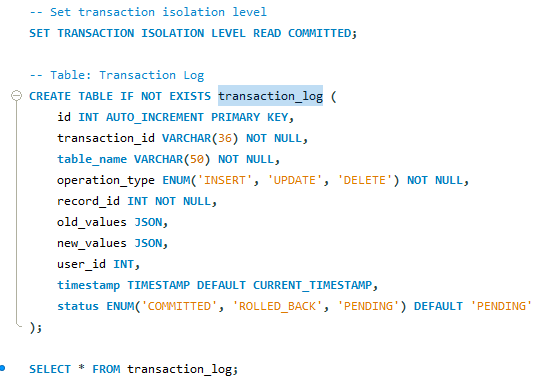
- Automatically recovers from system failures

- Restores medical data to a consistent state

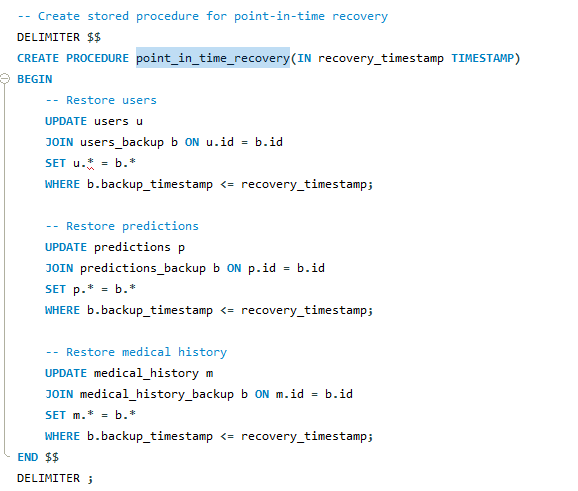
- Handles interrupted risk assessment calculations

- Maintains transaction atomicity

5. Implementation Example:



-- Point-in-Time Recovery Procedure



6. Recovery Steps:

a) System Failure Recovery:

- Check transaction log for incomplete transactions

- Restore database from last backup

- Replay committed transactions

- Verify medical data consistency

b) Data Corruption Recovery:

- Identify corrupted medical records

- Restore from backup

- Verify data integrity

- Update transaction logs

c) User Error Recovery:

- Identify incorrect updates

- Restore affected records

- Maintain audit trail

- Notify affected users

7. Security Measures:

- Encrypt backup files

- Secure transaction logs

- Implement access controls

- Maintain audit trails

8. Performance Considerations:

- Optimize backup procedures

- Manage storage efficiently

- Implement parallel recovery

- Monitor recovery times

This implementation ensures:

1. Data consistency in medical records

2. Reliable risk assessment history

3. Secure patient data management

4. Efficient recovery procedures

5. Comprehensive audit trailing

6. System availability

7. Data integrity protection

8. Patient safety

The system maintains:

- Medical data accuracy

- Risk assessment reliability

- Patient record integrity

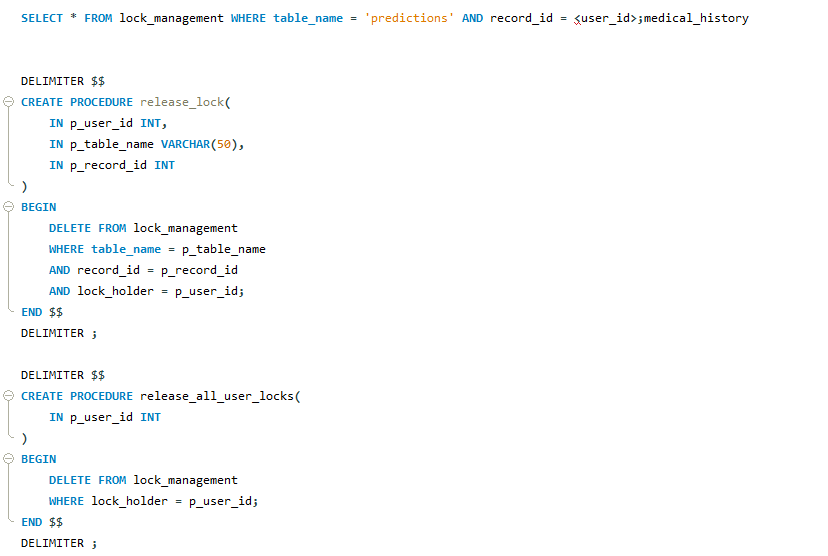
- System stability

- Data security

- Recovery efficiency

- Audit compliance

- Healthcare standardsstatements will be treated as part of a single unit of work. Transactions ensure data consistency by allowing a series of operations to be either committed (saved permanently) or rolled back (reverted) as a whole.



**CHAPTER-6 CODE FOR THE PROJECT**

## SQL CODE:

*-- Create the database*

CREATE DATABASE IF NOT EXISTS lung\_cancer\_db;

USE lung\_cancer\_db;

*-- Set transaction isolation level*

SET TRANSACTION ISOLATION LEVEL READ COMMITTED;

*-- Table: Transaction Log*

CREATE TABLE IF NOT EXISTS transaction\_log (

    id INT AUTO\_INCREMENT PRIMARY KEY,

    transaction\_id VARCHAR(36) NOT NULL,

    table\_name VARCHAR(50) NOT NULL,

    operation\_type ENUM('INSERT', 'UPDATE', 'DELETE') NOT NULL,

    record\_id INT NOT NULL,

    old\_values JSON,

    new\_values JSON,

    user\_id INT,

    timestamp TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

    status ENUM('COMMITTED', 'ROLLED\_BACK', 'PENDING') DEFAULT 'PENDING'

);

*-- Table: Version Control*

CREATE TABLE IF NOT EXISTS version\_control (

    id INT AUTO\_INCREMENT PRIMARY KEY,

    table\_name VARCHAR(50) NOT NULL,

    record\_id INT NOT NULL,

    version\_number INT NOT NULL DEFAULT 1,

    last\_modified TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,

    modified\_by INT,

    UNIQUE KEY (table\_name, record\_id)

);

*-- Table: Lock Management*

CREATE TABLE IF NOT EXISTS lock\_management (

    id INT AUTO\_INCREMENT PRIMARY KEY,

    table\_name VARCHAR(50) NOT NULL,

    record\_id INT NOT NULL,

    lock\_type ENUM('SHARED', 'EXCLUSIVE') NOT NULL,

    lock\_holder INT NOT NULL,

    lock\_timestamp TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

    lock\_timeout TIMESTAMP,

    UNIQUE KEY (table\_name, record\_id)

);

*-- Table: Users (for authentication)*

CREATE TABLE IF NOT EXISTS users (

    id INT AUTO\_INCREMENT PRIMARY KEY,

    name VARCHAR(100) NOT NULL,

    email VARCHAR(100) UNIQUE NOT NULL,

    password\_hash VARCHAR(255) NOT NULL,

    date\_of\_birth DATE,

    created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

    last\_login TIMESTAMP NULL,

    version INT DEFAULT 1

);

*-- Table: User Medical History*

CREATE TABLE IF NOT EXISTS medical\_history (

    id INT AUTO\_INCREMENT PRIMARY KEY,

    user\_id INT NOT NULL,

    family\_history\_of\_cancer ENUM('yes', 'no', 'unknown') DEFAULT 'unknown',

    years\_smoking INT DEFAULT 0,

    packs\_per\_day DECIMAL(3,1) DEFAULT 0.0,

    previous\_lung\_diseases TEXT,

    occupational\_exposure ENUM('yes', 'no', 'unknown') DEFAULT 'unknown',

    occupational\_details TEXT,

    updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,

    version INT DEFAULT 1,

    FOREIGN KEY (user\_id) REFERENCES users(id) ON DELETE CASCADE

);

*-- Table: Symptoms (stores detailed symptom information)*

CREATE TABLE IF NOT EXISTS symptoms (

    id INT AUTO\_INCREMENT PRIMARY KEY,

    name VARCHAR(100) NOT NULL,

    description TEXT,

    severity\_scale INT DEFAULT 3 COMMENT 'Scale from 1-5, with 5 being most severe',

    related\_to\_lung\_cancer BOOLEAN DEFAULT TRUE

);

*-- Table: Predictions (stores user symptoms and prediction results)*

CREATE TABLE IF NOT EXISTS predictions (

    id INT AUTO\_INCREMENT PRIMARY KEY,

    age INT CHECK (age BETWEEN 0 AND 120),

    gender ENUM('Male', 'Female', 'Other') NOT NULL,

    smoking ENUM('yes', 'no') NOT NULL,

    cough ENUM('yes', 'no') NOT NULL,

    chest\_pain ENUM('yes', 'no') NOT NULL,

    fatigue ENUM('yes', 'no') NOT NULL,

    shortness\_of\_breath ENUM('yes', 'no') NOT NULL,

    prediction VARCHAR(255) NOT NULL,

    risk\_score DECIMAL(5,2) DEFAULT 0.0,

    prediction\_date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

    version INT DEFAULT 1

);

*-- Table: User Predictions (links users to their predictions)*

CREATE TABLE IF NOT EXISTS user\_predictions (

    id INT AUTO\_INCREMENT PRIMARY KEY,

    user\_id INT,

    prediction\_id INT,

    notes TEXT,

    FOREIGN KEY (user\_id) REFERENCES users(id) ON DELETE CASCADE,

    FOREIGN KEY (prediction\_id) REFERENCES predictions(id) ON DELETE CASCADE

);

*-- Table: Medical Recommendations*

CREATE TABLE IF NOT EXISTS recommendations (

    id INT AUTO\_INCREMENT PRIMARY KEY,

    risk\_level ENUM('Low', 'Moderate', 'High') NOT NULL,

    recommendation\_text TEXT NOT NULL,

    resource\_links TEXT

);

*-- Table: User Feedback (for system improvement)*

CREATE TABLE IF NOT EXISTS user\_feedback (

    id INT AUTO\_INCREMENT PRIMARY KEY,

    user\_id INT,

    prediction\_id INT,

    feedback\_text TEXT NOT NULL,

    rating INT CHECK (rating BETWEEN 1 AND 5),

    created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

    FOREIGN KEY (user\_id) REFERENCES users(id) ON DELETE SET NULL,

    FOREIGN KEY (prediction\_id) REFERENCES predictions(id) ON DELETE SET NULL

);

*-- Table: Deleted Predictions Log*

CREATE TABLE IF NOT EXISTS deleted\_predictions\_log (

    id INT AUTO\_INCREMENT PRIMARY KEY,

    deleted\_prediction\_id INT,

    deleted\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

*-- Create backup table for critical data*

CREATE TABLE IF NOT EXISTS users\_backup LIKE users;

CREATE TABLE IF NOT EXISTS predictions\_backup LIKE predictions;

CREATE TABLE IF NOT EXISTS medical\_history\_backup LIKE medical\_history;

*-- Create stored procedure for backup*

DELIMITER $$

CREATE PROCEDURE create\_backup()

BEGIN

    DECLARE backup\_timestamp TIMESTAMP;

    SET backup\_timestamp = CURRENT\_TIMESTAMP;

*-- Backup users*

    INSERT INTO users\_backup

    SELECT \*, backup\_timestamp FROM users;

*-- Backup predictions*

    INSERT INTO predictions\_backup

    SELECT \*, backup\_timestamp FROM predictions;

*-- Backup medical history*

    INSERT INTO medical\_history\_backup

    SELECT \*, backup\_timestamp FROM medical\_history;

END $$

DELIMITER ;

*-- Create stored procedure for point-in-time recovery*

DELIMITER $$

CREATE PROCEDURE point\_in\_time\_recovery(IN recovery\_timestamp TIMESTAMP)

BEGIN

*-- Restore users*

    UPDATE users u

    JOIN users\_backup b ON u.id = b.id

    SET u.\* = b.\*

    WHERE b.backup\_timestamp <= recovery\_timestamp;

*-- Restore predictions*

    UPDATE predictions p

    JOIN predictions\_backup b ON p.id = b.id

    SET p.\* = b.\*

    WHERE b.backup\_timestamp <= recovery\_timestamp;

*-- Restore medical history*

    UPDATE medical\_history m

    JOIN medical\_history\_backup b ON m.id = b.id

    SET m.\* = b.\*

    WHERE b.backup\_timestamp <= recovery\_timestamp;

END $$

DELIMITER ;

*-- Create trigger for transaction logging*

DELIMITER $$

CREATE TRIGGER log\_user\_changes

AFTER UPDATE ON users

FOR EACH ROW

BEGIN

    INSERT INTO transaction\_log (transaction\_id, table\_name, operation\_type, record\_id, old\_values, new\_values)

    VALUES (

        UUID(),

        'users',

        'UPDATE',

        NEW.id,

        JSON\_OBJECT(

            'name', OLD.name,

            'email', OLD.email,

            'version', OLD.version

        ),

        JSON\_OBJECT(

            'name', NEW.name,

            'email', NEW.email,

            'version', NEW.version

        )

    );

END $$

DELIMITER ;

*-- Create stored procedure for deadlock detection*

DELIMITER $$

CREATE PROCEDURE detect\_deadlocks()

BEGIN

    DECLARE done INT DEFAULT 0;

    DECLARE lock\_id INT;

    DECLARE cur CURSOR FOR

        SELECT id FROM lock\_management

        WHERE lock\_timeout < CURRENT\_TIMESTAMP;

    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;

    OPEN cur;

    read\_loop: LOOP

        FETCH cur INTO lock\_id;

        IF done THEN

            LEAVE read\_loop;

        END IF;

*-- Release expired locks*

        DELETE FROM lock\_management WHERE id = lock\_id;

    END LOOP;

    CLOSE cur;

END $$

DELIMITER ;

*-- Create event scheduler for regular maintenance*

CREATE EVENT IF NOT EXISTS maintenance\_schedule

ON SCHEDULE EVERY 1 DAY

DO

BEGIN

*-- Create daily backup*

    CALL create\_backup();

*-- Clean up old transaction logs (keep last 30 days)*

    DELETE FROM transaction\_log

    WHERE timestamp < DATE\_SUB(CURRENT\_TIMESTAMP, INTERVAL 30 DAY);

*-- Check for deadlocks*

    CALL detect\_deadlocks();

END;

*-- Enable event scheduler*

SET GLOBAL event\_scheduler = ON;

*-- Insert sample data for testing*

*-- Users*

*-- INSERT INTO users (name, email, password\_hash, date\_of\_birth)*

*-- VALUES*

*-- ('John Doe', 'john@example.com', '$2b$12$ILv4xGHgz0DzFOEI8WtCre6GX6KtSQa/D1Yyz7.TqVw9AYyRlGK6y', '1980-05-15'),  -- Password: password123*

*-- ('Jane Smith', 'jane@example.com', '$2b$12$ILv4xGHgz0DzFOEI8WtCre6GX6KtSQa/D1Yyz7.TqVw9AYyRlGK6y', '1992-03-21');*

*-- Medical History*

*-- INSERT INTO medical\_history (user\_id, family\_history\_of\_cancer, years\_smoking, packs\_per\_day, previous\_lung\_diseases, occupational\_exposure)*

*-- VALUES*

*-- (1, 'yes', 15, 1.5, 'Chronic bronchitis', 'yes'),*

*-- (2, 'no', 0, 0.0, 'None', 'no');*

*-- Symptoms*

*-- INSERT INTO symptoms (name, description, severity\_scale, related\_to\_lung\_cancer)*

*-- VALUES*

*-- ('Persistent Cough', 'A cough that lasts for more than 2-3 weeks and doesn\'t improve', 4, TRUE),*

*-- ('Chest Pain', 'Pain in the chest area that may worsen with deep breathing or coughing', 5, TRUE),*

*-- ('Shortness of Breath', 'Feeling breathless or having difficulty breathing', 4, TRUE),*

*-- ('Fatigue', 'Feeling unusually tired or lacking energy', 3, TRUE),*

*-- ('Hoarseness', 'Changes in voice, such as hoarseness or raspiness', 2, TRUE),*

*-- ('Unexplained Weight Loss', 'Losing weight without trying', 5, TRUE);*

*-- Predictions*

*-- INSERT INTO predictions (age, gender, smoking, cough, chest\_pain, fatigue, shortness\_of\_breath, prediction, risk\_score)*

*-- VALUES*

*-- (45, 'Male', 'yes', 'yes', 'no', 'yes', 'yes', 'High risk of lung cancer', 12.5),*

*-- (32, 'Female', 'no', 'yes', 'no', 'no', 'no', 'Low risk of lung cancer', 3.2),*

*-- (67, 'Male', 'yes', 'yes', 'yes', 'yes', 'yes', 'High risk of lung cancer', 15.8),*

*-- (28, 'Female', 'no', 'no', 'no', 'yes', 'no', 'Low risk of lung cancer', 2.1);*

*-- Link users to predictions*

*-- INSERT INTO user\_predictions (user\_id, prediction\_id, notes)*

*-- VALUES*

*-- (1, 1, 'Initial assessment after developing cough'),*

*-- (1, 3, 'Follow-up after chest pain developed'),*

*-- (2, 2, 'Routine checkup');*

*-- Recommendations*

*-- INSERT INTO recommendations (risk\_level, recommendation\_text, resource\_links)*

*-- VALUES*

*-- ('Low', 'Maintain a healthy lifestyle. If you smoke, consider quitting. Schedule regular check-ups with your doctor.', 'https://www.cancer.org/healthy/stay-away-from-tobacco.html'),*

*-- ('Moderate', 'Schedule an appointment with your doctor to discuss your symptoms. Consider a chest X-ray or CT scan for further evaluation.', 'https://www.cancer.org/cancer/lung-cancer/detection-diagnosis-staging.html'),*

*-- ('High', 'Contact your doctor immediately to schedule comprehensive testing. This may include imaging tests and possibly a biopsy.', 'https://www.cancer.org/cancer/lung-cancer/treating.html');*

*-- User Feedback*

*-- INSERT INTO user\_feedback (user\_id, prediction\_id, feedback\_text, rating)*

*-- VALUES*

*-- (1, 1, 'The assessment was helpful and matched what my doctor told me.', 5),*

*-- (2, 2, 'I would have liked more detailed recommendations.', 3);*

SELECT \* from users;

SELECT \* FROM user\_predictions;

SHOW TABLES;

DROP TABLE IF EXISTS user\_prediction;

DROP TABLE IF EXISTS prediction;

DROP TABLE IF EXISTS deleted\_predictions;

CREATE OR REPLACE VIEW user\_prediction\_history AS

SELECT

    u.id AS user\_id,

    u.name AS user\_name,

    u.email AS user\_email,

    p.id AS prediction\_id,

    p.age, p.gender, p.smoking, p.cough, p.chest\_pain, p.fatigue, p.shortness\_of\_breath,

    p.prediction, p.risk\_score, p.prediction\_date

FROM users u

JOIN user\_predictions up ON u.id = up.user\_id

JOIN predictions p ON up.prediction\_id = p.id

ORDER BY p.prediction\_date DESC;

DELIMITER $$

CREATE TRIGGER after\_prediction\_delete

AFTER DELETE ON predictions

FOR EACH ROW

BEGIN

    INSERT INTO deleted\_predictions\_log (deleted\_prediction\_id) VALUES (OLD.id);

END $$

DELIMITER ;

DELIMITER $$

CREATE PROCEDURE update\_all\_risk\_scores()

BEGIN

    DECLARE done INT DEFAULT 0;

    DECLARE pred\_id INT;

    DECLARE cur CURSOR FOR SELECT id FROM predictions;

    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;

    OPEN cur;

    read\_loop: LOOP

        FETCH cur INTO pred\_id;

        IF done THEN

            LEAVE read\_loop;

        END IF;

*-- Example: set risk\_score to 0 for demonstration*

        UPDATE predictions SET risk\_score = 0 WHERE id = pred\_id;

    END LOOP;

    CLOSE cur;

END $$

DELIMITER ;

SELECT \* FROM lock\_management;

SELECT \* FROM lock\_management WHERE table\_name = 'predictions' AND record\_id = <user\_id>;

## Employee\_Adding:

package rest;

import java.awt.EventQueue;

import javax.swing.JFrame; import javax.swing.JLabel; import java.awt.Font;

import java.awt.event.ActionEvent; import java.awt.event.ActionListener; import java.sql.Connection;

import java.sql.DriverManager; import java.sql.PreparedStatement; import java.util.Scanner;

import javax.swing.JTextField; import javax.swing.JButton;

public class Employee\_add { String field1,field3,field4; int field2;

JFrame frame2;

private JTextField textField; private JTextField textField\_1; private JTextField textField\_2; private JTextField textField\_3;

public static void main(String[] args) { EventQueue.invokeLater(new Runnable() {

public void run() { try {

Employee\_add window = new Employee\_add(); window.frame2.setVisible(true);

} catch (Exception e) {

e.printStackTrace();

}

}

});

}

public Employee\_add() { initialize();

}

private void initialize() {

frame2 = new JFrame(); frame2.setBounds(100, 100, 450, 300);

frame2.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE); frame2.getContentPane().setLayout(null);

JLabel lblNewLabel = new JLabel("Gender");

lblNewLabel.setFont(new Font("Tahoma", Font.PLAIN, 15)); lblNewLabel.setBounds(30, 29, 98, 38); frame2.getContentPane().add(lblNewLabel);

JLabel lblSalary = new JLabel("Salary"); lblSalary.setFont(new Font("Tahoma", Font.PLAIN, 15)); lblSalary.setBounds(30, 77, 98, 38); frame2.getContentPane().add(lblSalary);

JLabel lblName = new JLabel("Name"); lblName.setFont(new Font("Tahoma", Font.PLAIN, 15)); lblName.setBounds(30, 125, 98, 38); frame2.getContentPane().add(lblName);

JLabel lblEmployeeid = new JLabel("Employee\_ID"); lblEmployeeid.setFont(new Font("Tahoma", Font.PLAIN, 14)); lblEmployeeid.setBounds(30, 173, 98, 38); frame2.getContentPane().add(lblEmployeeid);

textField = new JTextField(); textField.setBounds(160, 29, 145, 29); frame2.getContentPane().add(textField); textField.setColumns(10);

textField\_1 = new JTextField(); textField\_1.setColumns(10); textField\_1.setBounds(160, 77, 145, 29); frame2.getContentPane().add(textField\_1);

textField\_2 = new JTextField(); textField\_2.setColumns(10); textField\_2.setBounds(160, 125, 145, 29); frame2.getContentPane().add(textField\_2);

textField\_3 = new JTextField(); textField\_3.setColumns(10); textField\_3.setBounds(160, 180, 145, 29); frame2.getContentPane().add(textField\_3);

JButton btnNewButton = new JButton("Next"); btnNewButton.setFont(new Font("Tahoma", Font.PLAIN, 15)); btnNewButton.setBounds(297, 219, 109, 34); btnNewButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) { try {

Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/restaurant", "root", "teamosekire");

Scanner scan=new Scanner(System.in); field1=textField.getText();

field2=Integer.parseInt(textField\_1.getText()); field3=textField\_2.getText(); field4=textField\_3.getText();

String mini="insert into Employee values(?,?,?,?)"; PreparedStatement s3= con.prepareStatement(mini); s3.setString(1,field1);

s3.setInt(2, field2); s3.setString(3, field3); s3.setString(4, field4); s3.executeUpdate(); s3.close();

con.close();

Employee\_D obj = new Employee\_D(); obj.frame1.setVisible(true); frame2.dispose();

} catch (Exception e1) {

System.out.println(e1.getMessage());

}

}

});

frame2.getContentPane().add(btnNewButton);

}

}

## Employee Display:

package rest;

import java.awt.BorderLayout; import java.awt.Container; import java.awt.EventQueue;

import java.awt.event.ActionEvent; import java.awt.event.ActionListener; import java.sql.Connection;

import java.sql.DriverManager; import java.sql.PreparedStatement; import java.sql.ResultSet;

import javax.swing.JButton; import javax.swing.JFrame; import javax.swing.JScrollPane; import javax.swing.JTable;

import javax.swing.table.DefaultTableModel;

public class Employee\_D {

JFrame frame1; private JTable jtbl;

private DefaultTableModel model;

/\*\*

\* Launch the application.

\*/

public static void main(String[] args) { EventQueue.invokeLater(new Runnable() {

public void run() { try {

Employee\_D window = new Employee\_D(); window.frame1.setVisible(true);

} catch (Exception e) {

e.printStackTrace();

}

}

});

}

/\*\*

\* Create the application.

\*/

public Employee\_D() {

initialize();

}

/\*\*

\* Initialize the contents of the frame.

\*/

private void initialize() {

frame1 = new JFrame(); frame1.setBounds(100, 100, 450, 300);

frame1.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

model = new DefaultTableModel(); model.addColumn("Name"); model.addColumn("Gender"); model.addColumn("Salary"); model.addColumn("Employee\_id");

jtbl = new JTable(model); frame1.getContentPane().setLayout(new BorderLayout());

frame1.getContentPane().add(new JScrollPane(jtbl), BorderLayout.CENTER);

JButton btnLoadData = new JButton("Back"); btnLoadData.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) { MainPage1 obj = new MainPage1(); obj.frame.setVisible(true); frame1.dispose();

}

});

frame1.getContentPane().add(btnLoadData, BorderLayout.SOUTH); try {

Class.forName("com.mysql.cj.jdbc.Driver"); Connection con =

DriverManager.getConnection("jdbc:mysql://localhost:3306/restaurant", "root", "teamosekire");

String query = "SELECT \* FROM employee"; PreparedStatement s1 = con.prepareStatement(query); ResultSet rs = s1.executeQuery();

while (rs.next()) {

model.addRow(new Object[]{rs.getString("Name"), rs.getString("Gender"), rs.getString("Salary"), rs.getString("Employee\_id")});

}

con.close();

s1.close();

rs.close();

} catch (Exception e) {

System.out.println(e.getMessage());

}

frame1.pack();

}

}

## Menu Display:

package rest;

import java.awt.BorderLayout; import java.awt.EventQueue; import java.awt.event.ActionEvent;

import java.awt.event.ActionListener; import java.sql.Connection;

import java.sql.DriverManager; import java.sql.PreparedStatement; import java.sql.ResultSet;

import javax.swing.JButton; import javax.swing.JFrame; import javax.swing.JScrollPane; import javax.swing.JTable;

import javax.swing.table.DefaultTableModel; public class Menu {

JFrame frame1;

private JTable jtbl;

private DefaultTableModel model;

public static void main(String[] args) { EventQueue.invokeLater(new Runnable() {

public void run() { try {

Menu window = new Menu(); window.frame1.setVisible(true);

} catch (Exception e) {

e.printStackTrace();

}

}

});

}

public Menu() {

initialize();

}

private void initialize() {

frame1 = new JFrame(); frame1.setBounds(100, 100, 450, 300);

frame1.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

model = new DefaultTableModel(); model.addColumn("Timing"); model.addColumn("Cuisine"); model.addColumn("Price"); model.addColumn("Food\_ID");

jtbl = new JTable(model); frame1.getContentPane().setLayout(new BorderLayout());

frame1.getContentPane().add(new JScrollPane(jtbl), BorderLayout.CENTER);

JButton btnBack = new JButton("Back"); btnBack.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) { MainPage1 obj = new MainPage1(); obj.frame.setVisible(true); frame1.dispose();

}

});

frame1.getContentPane().add(btnBack, BorderLayout.SOUTH); try {

Class.forName("com.mysql.cj.jdbc.Driver"); Connection con =

DriverManager.getConnection("jdbc:mysql://localhost:3306/restaurant", "root", "teamosekire");

String query = "SELECT \* FROM Menu"; PreparedStatement s1 = con.prepareStatement(query); ResultSet rs = s1.executeQuery();

while (rs.next()) {

model.addRow(new Object[]{rs.getString("Timing"), rs.getString("Cuisine"), rs.getString("Price"), rs.getString("Food\_ID")});

}

con.close();

s1.close();

rs.close();

} catch (Exception e) {

System.out.println(e.getMessage());

}

frame1.pack();

}

}

## Billing:

package rest;

import java.awt.EventQueue; import java.awt.Image;

import java.awt.event.ActionEvent; import java.awt.event.ActionListener; import java.sql.Connection;

import java.sql.DriverManager; import java.sql.PreparedStatement; import java.sql.ResultSet;

import java.time.Instant; import java.util.Scanner;

import javax.swing.ImageIcon; import javax.swing.JFrame; import javax.swing.JLabel; import javax.swing.JTextField; import javax.swing.JButton;

public class Billing {

int field1,field2,field3,field4,field5,field6,field7; int n,id,Quantity;

String cost,service,tax; JFrame frame1;

private JTextField textField; private JTextField textField\_1;

private JTextField textField\_2; private JTextField textField\_3; private JTextField textField\_4; private JTextField textField\_5; private JTextField textField\_6; private JTextField textField\_7; private JTextField textField\_8; private JTextField textField\_9; private JTextField textField\_10; private JTextField textField\_11; private JTextField textField\_12; private JButton btnNewButton\_1;

public static void main(String[] args) { EventQueue.invokeLater(new Runnable() {

public void run() { try {

Billing window = new Billing(); window.frame1.setVisible(true);

} catch (Exception e) {

e.printStackTrace();

}

}

});

}

public Billing() {

initialize();

}

private void initialize() {

frame1 = new JFrame(); frame1.setBounds(100, 100, 1024, 540);

frame1.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

Image img= new ImageIcon(this.getClass().getResource("/Bill.jpg")).getImage(); frame1.getContentPane().setLayout(null);

textField = new JTextField(); textField.setBounds(214, 113, 113, 34); frame1.getContentPane().add(textField); textField.setColumns(10);

textField\_1 = new JTextField(); textField\_1.setColumns(10); textField\_1.setBounds(214, 160, 113, 34); frame1.getContentPane().add(textField\_1);

textField\_2 = new JTextField(); textField\_2.setColumns(10);

textField\_2.setBounds(214, 209, 113, 34); frame1.getContentPane().add(textField\_2);

textField\_3 = new JTextField(); textField\_3.setColumns(10); textField\_3.setBounds(214, 257, 113, 34); frame1.getContentPane().add(textField\_3);

textField\_4 = new JTextField(); textField\_4.setColumns(10); textField\_4.setBounds(214, 353, 113, 34); frame1.getContentPane().add(textField\_4);

textField\_5 = new JTextField(); textField\_5.setColumns(10); textField\_5.setBounds(214, 401, 113, 34); frame1.getContentPane().add(textField\_5);

textField\_6 = new JTextField(); textField\_6.setColumns(10); textField\_6.setBounds(214, 305, 113, 34); frame1.getContentPane().add(textField\_6);

textField\_7 = new JTextField(); textField\_7.setBounds(773, 115, 210, 40); frame1.getContentPane().add(textField\_7); textField\_7.setColumns(10);

textField\_8 = new JTextField(); textField\_8.setColumns(10); textField\_8.setBounds(773, 169, 210, 40); frame1.getContentPane().add(textField\_8);

textField\_9 = new JTextField(); textField\_9.setColumns(10); textField\_9.setBounds(773, 261, 210, 40); frame1.getContentPane().add(textField\_9);

textField\_10 = new JTextField(); textField\_10.setColumns(10); textField\_10.setBounds(773, 305, 210, 40); frame1.getContentPane().add(textField\_10);

textField\_11 = new JTextField(); textField\_11.setColumns(10); textField\_11.setBounds(773, 351, 210, 40); frame1.getContentPane().add(textField\_11);

textField\_12 = new JTextField(); textField\_12.setColumns(10);

textField\_12.setBounds(773, 216, 210, 40); frame1.getContentPane().add(textField\_12);

JButton btnNewButton = new JButton("Calculate"); btnNewButton.setBounds(8, 447, 174, 46); btnNewButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) { try {

Class.forName("com.mysql.cj.jdbc.Driver"); Connection con =

DriverManager.getConnection("jdbc:mysql://localhost:3306/restaurant", "root", "teamosekire");

Scanner scan=new Scanner(System.in); field1=Integer.parseInt(textField.getText()); field2=Integer.parseInt(textField\_1.getText()); field3=Integer.parseInt(textField\_2.getText()); field4=Integer.parseInt(textField\_3.getText()); field5=Integer.parseInt(textField\_6.getText()); field6=Integer.parseInt(textField\_4.getText()); field7=Integer.parseInt(textField\_5.getText());

n = (field1 \* 30) + (field2 \* 45) + (field3 \* 55) + (field4 \* 20) + (field5 \* 50) + (field6 \* 15) + (field7 \* 20);

Quantity=field1+field2+field3+field4+field5+field6+field7;

cost="BL"; id=456;

id++;

service = cost + String.valueOf(id);

Instant t1=java.time.Clock.systemUTC().instant(); String time = t1.toString(); textField\_7.setText(service); textField\_8.setText(time); textField\_12.setText(String.valueOf(n)); textField\_9.setText(String.valueOf(100)); textField\_10.setText(String.valueOf(n\*0.12));

textField\_11.setText(String.valueOf(n+(n\*0.12)+100)); String mini="insert into Billing values(?,?,?,?)"; PreparedStatement s3= con.prepareStatement(mini); s3.setString(1,service);

s3.setInt(2, Quantity); s3.setString(3, time);

s3.setInt(4, (int) (n+(n\*0.12)+100)); s3.executeUpdate();

s3.close();

scan.close();

con.close();

} catch (Exception e1) {

System.out.println(e1.getMessage());

}

}

});

frame1.getContentPane().add(btnNewButton);

JButton btnNewButton\_2 = new JButton("Clear"); btnNewButton\_2.setBounds(202, 447, 133, 46); btnNewButton\_2.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) { textField.setText(""); textField\_1.setText(""); textField\_2.setText(""); textField\_3.setText(""); textField\_4.setText(""); textField\_5.setText(""); textField\_6.setText(""); textField\_7.setText(""); textField\_8.setText(""); textField\_9.setText(""); textField\_10.setText(""); textField\_11.setText(""); textField\_12.setText("");

}

});

frame1.getContentPane().add(btnNewButton\_2);

JButton btnNewButton\_2\_1 = new JButton("Back"); btnNewButton\_2\_1.setBounds(354, 447, 127, 46); btnNewButton\_2\_1.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) { MainPage1 obj = new MainPage1(); obj.frame.setVisible(true); frame1.dispose();

}

});

frame1.getContentPane().add(btnNewButton\_2\_1); JLabel label\_1 = new JLabel(""); label\_1.setBounds(0, 0, 1010, 503); label\_1.setIcon(new ImageIcon(img)); frame1.getContentPane().add(label\_1); }

}

## Account Display:

package rest;

import java.awt.BorderLayout; import java.awt.EventQueue; import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.sql.Connection; import java.sql.DriverManager;

import java.sql.PreparedStatement; import java.sql.ResultSet;

import javax.swing.JButton; import javax.swing.JFrame; import javax.swing.JScrollPane; import javax.swing.JTable;

import javax.swing.table.DefaultTableModel; public class Account {

JFrame frame1; private JTable jtbl;

private DefaultTableModel model; public static void main(String[] args) {

EventQueue.invokeLater(new Runnable() { public void run() {

try {

Account window = new Account(); window.frame1.setVisible(true);

} catch (Exception e) {

e.printStackTrace();

}

}

});

}

public Account() {

initialize();

}

private void initialize() {

frame1 = new JFrame(); frame1.setBounds(100, 100, 450, 300);

frame1.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

model = new DefaultTableModel(); model.addColumn("Bill\_ID"); model.addColumn("Total\_count"); model.addColumn("Handled\_By"); model.addColumn("Table\_ID");

jtbl = new JTable(model); frame1.getContentPane().setLayout(new BorderLayout());

frame1.getContentPane().add(new JScrollPane(jtbl), BorderLayout.CENTER);

JButton btnBack = new JButton("Back"); btnBack.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) { MainPage1 obj = new MainPage1(); obj.frame.setVisible(true); frame1.dispose();

}

});

frame1.getContentPane().add(btnBack, BorderLayout.SOUTH); try {

Class.forName("com.mysql.cj.jdbc.Driver"); Connection con =

DriverManager.getConnection("jdbc:mysql://localhost:3306/restaurant", "root", "teamosekire");

String query = "SELECT \* FROM BILL"; PreparedStatement s1 = con.prepareStatement(query); ResultSet rs = s1.executeQuery();

while (rs.next()) {

model.addRow(new Object[]{rs.getString("Bill\_ID"), rs.getString("Total\_count"), rs.getString("Handled\_By"), rs.getString("Table\_ID")});

}

con.close();

s1.close();

rs.close();

} catch (Exception e) {

System.out.println(e.getMessage());

}

frame1.pack();

}

}

**CHAPTER-7 RESULT AND DISCUSSION**

The Restaurant Management System is an essential component of efficient restaurant operations, ensuring smooth customer service and streamlined processes. This system encompasses various modules to handle customer data, menu management, order processing, billing calculations, and payment processing.

**ADVERTISMENTS Table:** This table stores information about advertisements, including the deadline, advertisement area, theme, budget, and customer ID.

**BILL Table:** This table tracks bills with a unique bill ID, total count, the person who handled the bill, and a table ID.In the billing system, this table could be used to store information about generated bills, their handling, and associated details.

**CASHIER Table:** The cashier table contains data about cashiers, including their admin ID, associated bill ID, food items, customer name, and price.In the billing system, this table might be used to track cashier activities related to bill payments or other transactions.

**CHEF Table:** This table holds information about chefs, including their gender, date of birth, name, salary, and chef ID.

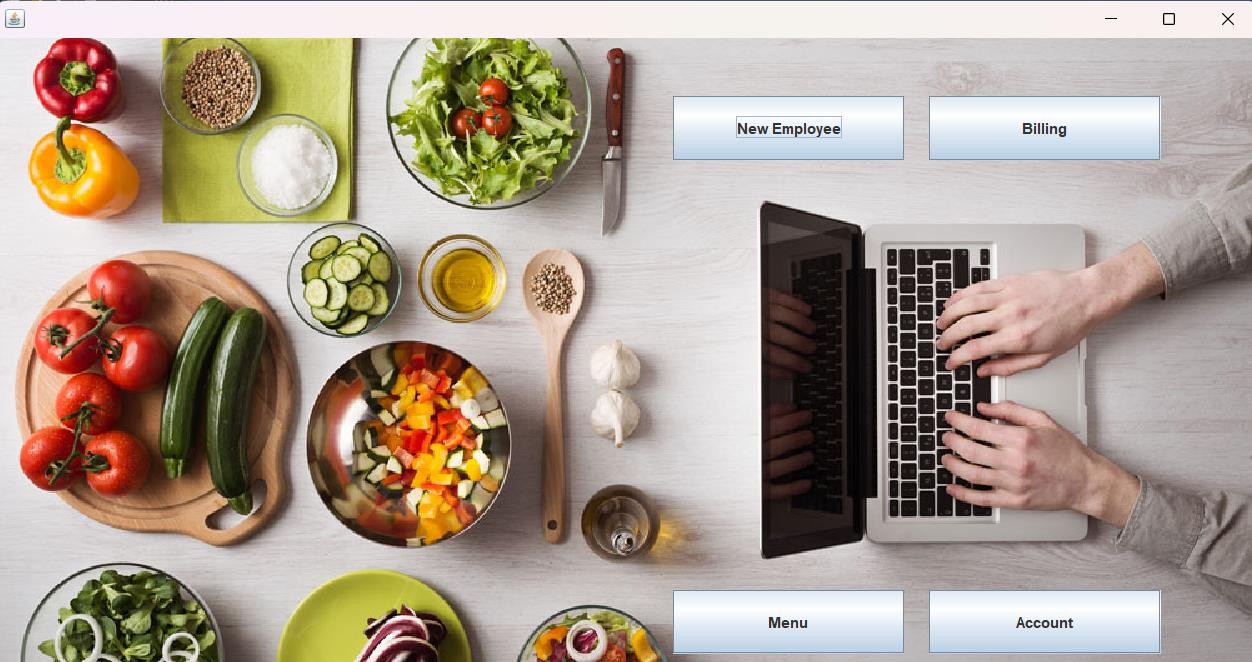
**FEEDBACK Table:** This table records feedback from customers regarding food quality, service, favorite food, improvement suggestions, and customer ID.In the billing system context, feedback mechanisms might be used for customer satisfaction surveys or service improvement initiatives..

**FOOD\_DELIVERY Table:** This table contains information about food deliveries, including customer ID, total price, items details, delivery partner, and branch ID.

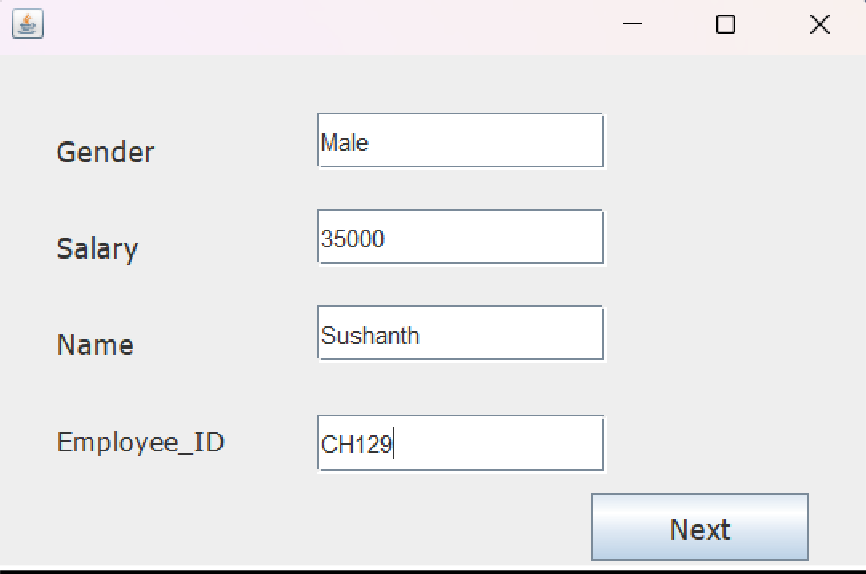
**Database Management:** The system maintains a robust database architecture for data storage, retrieval, and backup. It ensures data integrity, security, and scalability to handle large volumes of customer information and transactional data effectively.

In summary, the Restaurant Management System streamlines restaurant operations, enhances customer satisfaction, and supports data-driven decision-making through efficient database management and reporting capabilities.

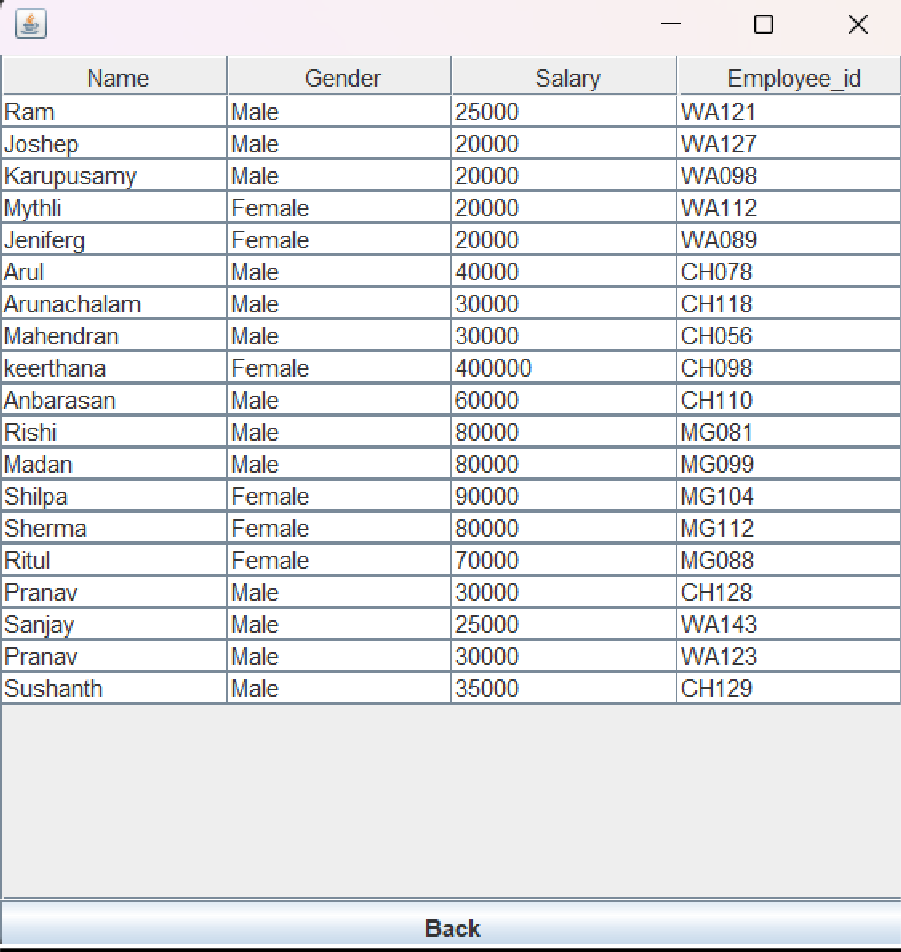
**Main Page:-**

****

**New Employee:-**

****

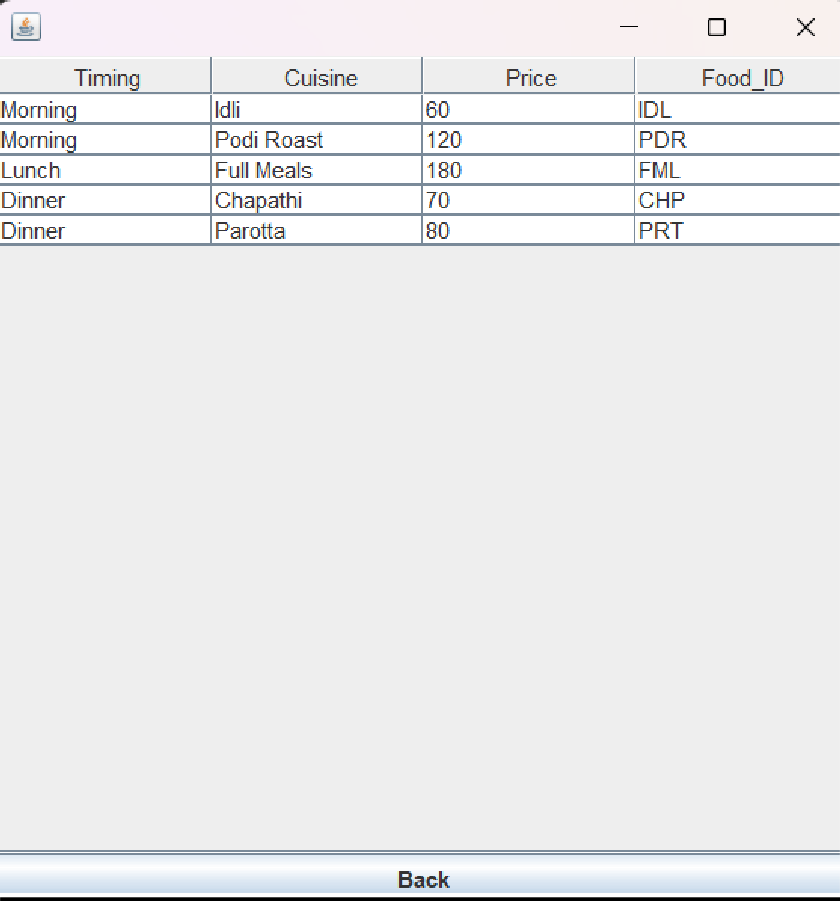
**Employee Display: -**



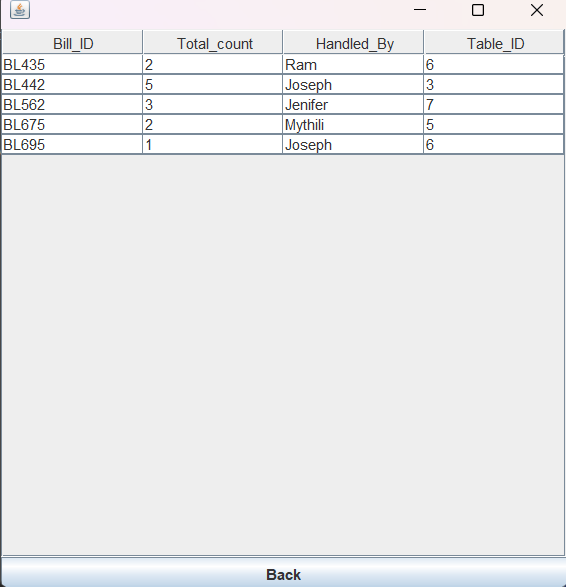
**Billing:-**

****

**Menu:-**



**Account Display:-**

****

**FUTURE SCOPE AND LIMITATIONS**

Software developers may not expect. The following principles enhances extensibility like hide data structure, avoid traversing multiple.

Links or methods avoid case statements on object type and distinguish public and private operations.

* Reusability: Reusability is possible as and when require in this application. We can update it next version. Reusable software reduces design, coding and testing cost by amortizing effort

Over several designs. Reducing the amount of code also simplifies understanding, which increases the likelihood that the code is correct. We follow up both types of reusability:

Sharing of newly written code within a project and reuse of previously written code on new projects.

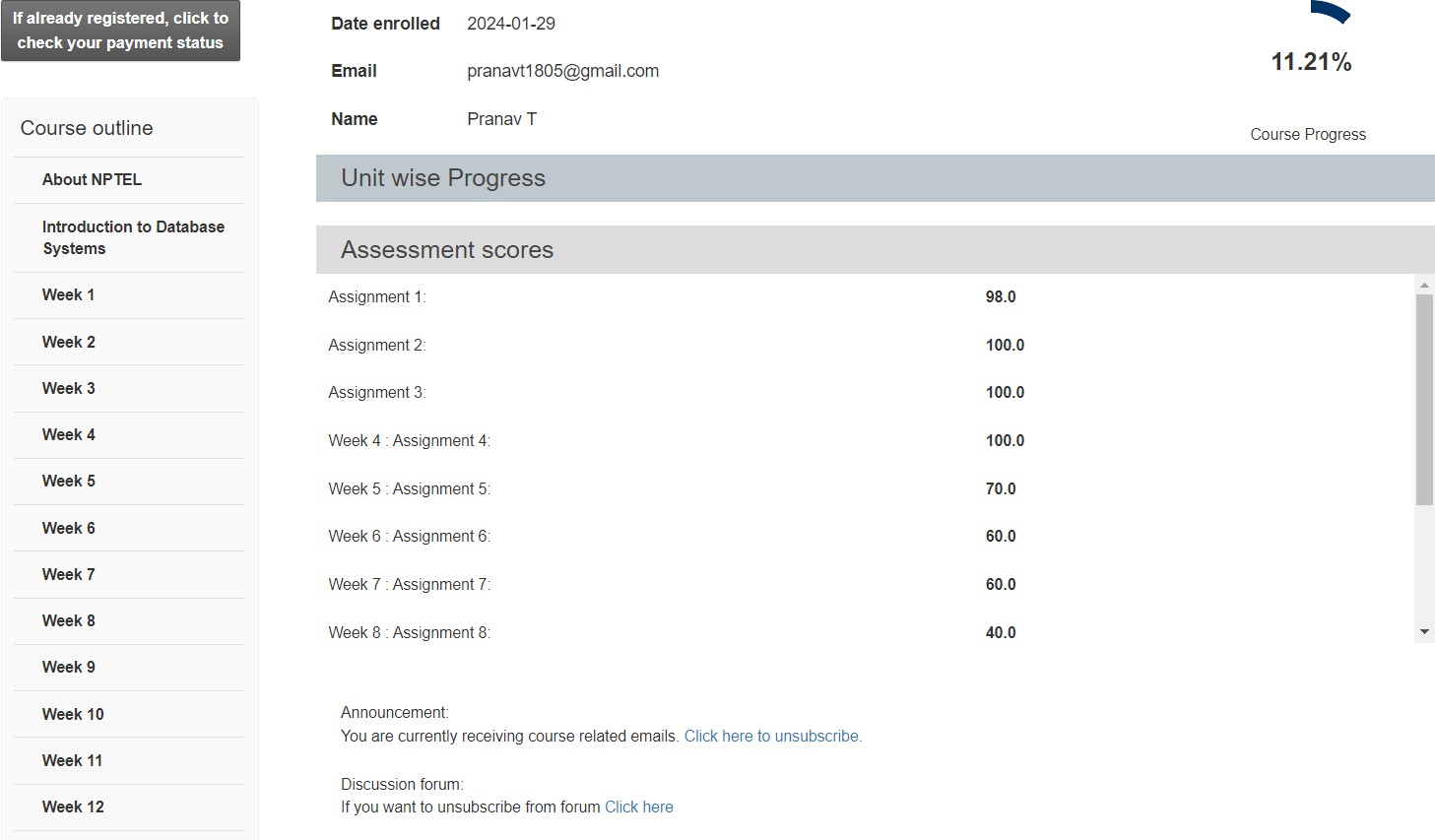
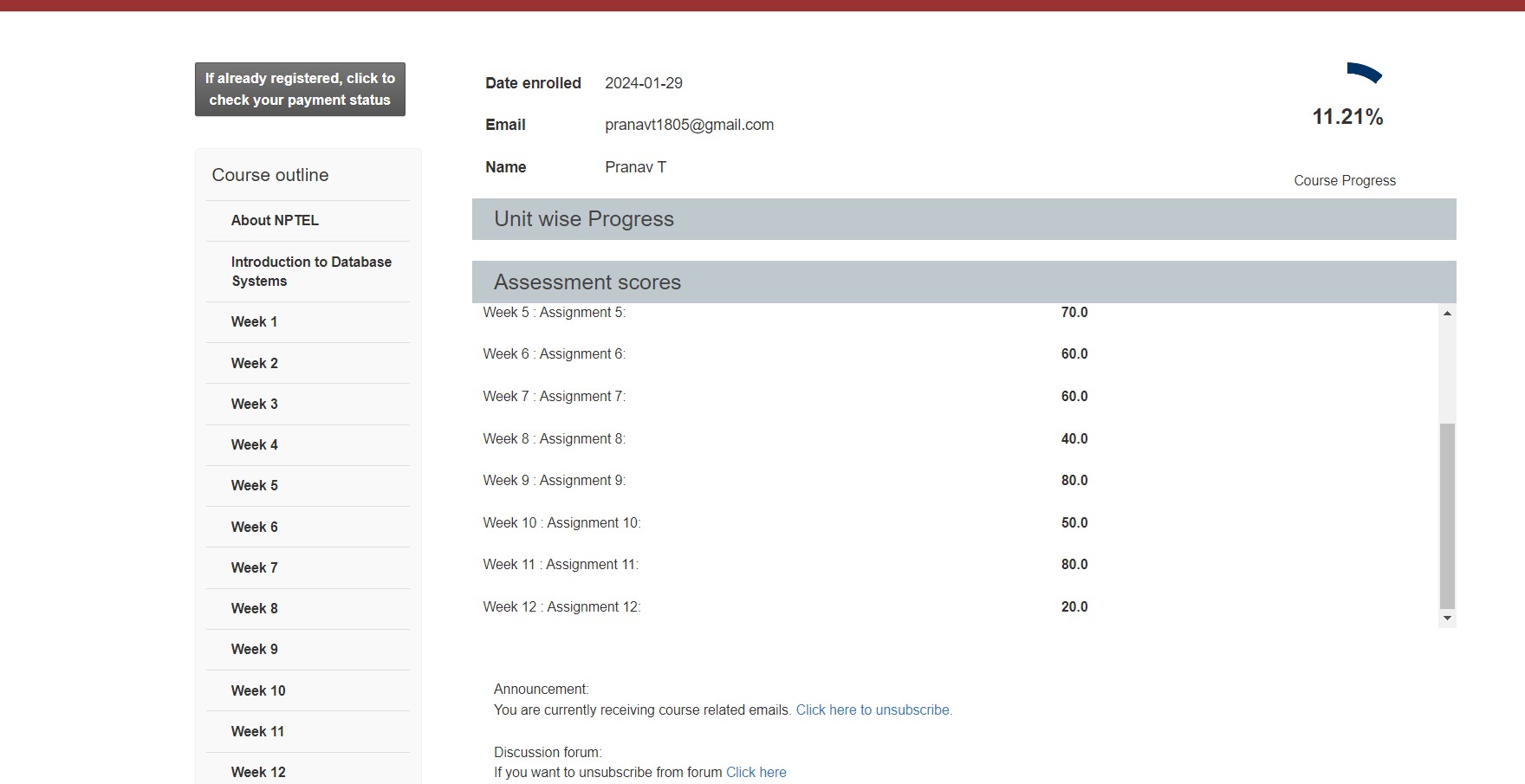
* Understand ability: A method is understandable if someone other than the creator of the method can understand the code (as well as the creator after a time lapse). We use the method, which small and coherent helps to accomplish this.
* Cost-effectiveness: Its cost is under the budget and make within given time period. It is desirable to aim for a system with a minimum cost subject to the condition that it must satisfy the entire requirement. Scope of this document is to put down the requirements, clearly identifying the information needed by the user, the source of the information and outputs expected from the system.

### LIMITATIONS:-

* + This application cannot be accessed remotely.
  + This application requires knowledgeable person to use this application.
  + This application does not have journals

**CONCLUSION**

After all the hard work is done for the Restaurant Management System, it is a software designed to streamline restaurant operations, including managing billing cycles, paying bills, and overseeing various operational details. This software significantly reduces the need for manual data entry, leading to greater efficiency in restaurant management. Its user-friendly interface makes it easy for anyone to use, thereby decreasing the time required for administrative tasks and enhancing overall productivity.



NPTEL

(Signature of Candidate)

|  |  |  |
| --- | --- | --- |
| MORIG SESSIO (Ï) | National Programme on Technology Enhanced Learning |  |
| Hall Ticket For | 2024 Apr: CS55 Introduction to Database Systems - Online |

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Candidate Name | Pranav T | | | | | | | | | |  |
| Roll No | NOC24CS55S553400299 | | | | | Seating Number | | | 53400299 | |
| Date of Birth | 18-05-2005 | | | | | | | | | |
| PwD Status | No | Compensatory Time Required | | | N.A | | Scribe Required | | | N.A |
| Exam Date | Sunday, 21 April, 2024 | | | | | | | | | |
| Reporting Time | 08:00 am | | | Gate Closure | | | | 09:30 am | | |
| Exam Timing | 09:00 am | | | Shift | | | | FN | | |
| Test Centre Name | iON Digital Zone iDZ Kovilambakkam | | | | | | | | | |  |
| Test Centre Address | Fortune Towers,1st and 2nd Floor,SH 109, 200 Feet Thoraipakkam Pallavaram Radial Road, Near Eachangadu signal,Kovilambakkam, Chennai, Tamil Nadu, India - 600117 | | | | | | | | | |
| **NPTEL Coordinator** | | |  | | | | | | | | |

|  |
| --- |
| **NPTEL EXAM - 21 APRIL, 2024**  **General instructions for candidates - FN**  (All timings mentioned here are in IST) |
| **The total duration of the examination is 180 minutes.**  **Candidates will be permitted to leave the examination hall only after 10:30 am, on a need basis.** |
| **Hall ticket and Entry:**   1. The Hall Ticket must be presented for verification along with one original photo identification (not photocopy or scanned copy). Examples of acceptable photo identification documents are School ID, College ID, Employee ID, Driving License, Passport, PAN card, Voter ID, and Aadhaar-ID. A printed copy of the hall ticket and original photo ID card should be brought to the exam centre. Hall ticket and ID card copies on the phone will not be permitted. 2. This Hall Ticket is valid only if the candidate’s photograph and signature images are legible. To ensure this, print the Hall Ticket on A4-sized paper using a laser printer, preferably a color photo printer. 3. **TIMELINE:** 8:00 am - Report to the examination venue | 8:40 am – Candidates will be permitted to occupy their allotted seats| 8:50 am – Candidates can login and start reading instructions prior to the examination | 9:00 am - Exam starts |   9:30 am - Gate closes, candidates will not be allowed after this time | 10:30 am Submit button will be enabled | 12:00 pm exam ends.   1. Candidates will be permitted to appear for the examination ONLY after their credentials are verified by center officials.   **P.T.O.** |

1. Candidates are advised to locate the examination center at least a day prior to the examination, so that they can reach the center on time for the examination.

**STATIONERY REQUIREMENTS:**

* + A4 sheets will be provided to candidates for rough work. Candidates have to write their name and registration number on the A4 Sheets before they start using it. The A4 sheets must be returned to the invigilator at the end of the examination.
  + On-screen calculator will be available during the exam. Candidates are advised to familiarize themselves with this virtual Scientific calculator well ahead of the exam.

Link: [https://www.tcsion.com/OnlineAssessment/ScientificCalculator/Calculator.html](http://www.tcsion.com/OnlineAssessment/ScientificCalculator/Calculator.html)

* + You should bring your own pen/pencil; it will NOT be given at the examination centre.

**DRESS CODE:**

* + Candidates are expected to come in professional attire to write the exams.
  + Candidates wearing SHORTS will NOT be permitted inside the exam hall.

**PERMITTED:**

* + You may bring vehicle keys inside the exam hall.
  + You are advised to carry your own drinking water in a transparent bottle.
  + Candidates are allowed to bring sanitizer in a small transparent bottle.

**NOT PERMITTED:**

* + Watches, wallets, mobile phones, Bluetooth devices, microphones, pagers, health bands or any other electronic gadgets, any printed/blank/handwritten paper, log tables, writing pads, scales, geometry/pencil-boxes, pouches, calculators, pen drives, electronic pens, handbags, goggles, electronic vehicle keys or similar such items are NOT allowed inside the examination centre. There may not be any facility for the safekeeping of these devices outside the examination hall; it will be prudent not to bring valuables to the examination center. Candidates will not be permitted to carry any food items in the exam centre. We suggest that you bring a bag to keep routine belongings outside the exam hall. Neither NPTEL nor the exam provider takes responsibility for the bag and the belongings. You may keep it outside at your own risk.

**MANDATORY :**

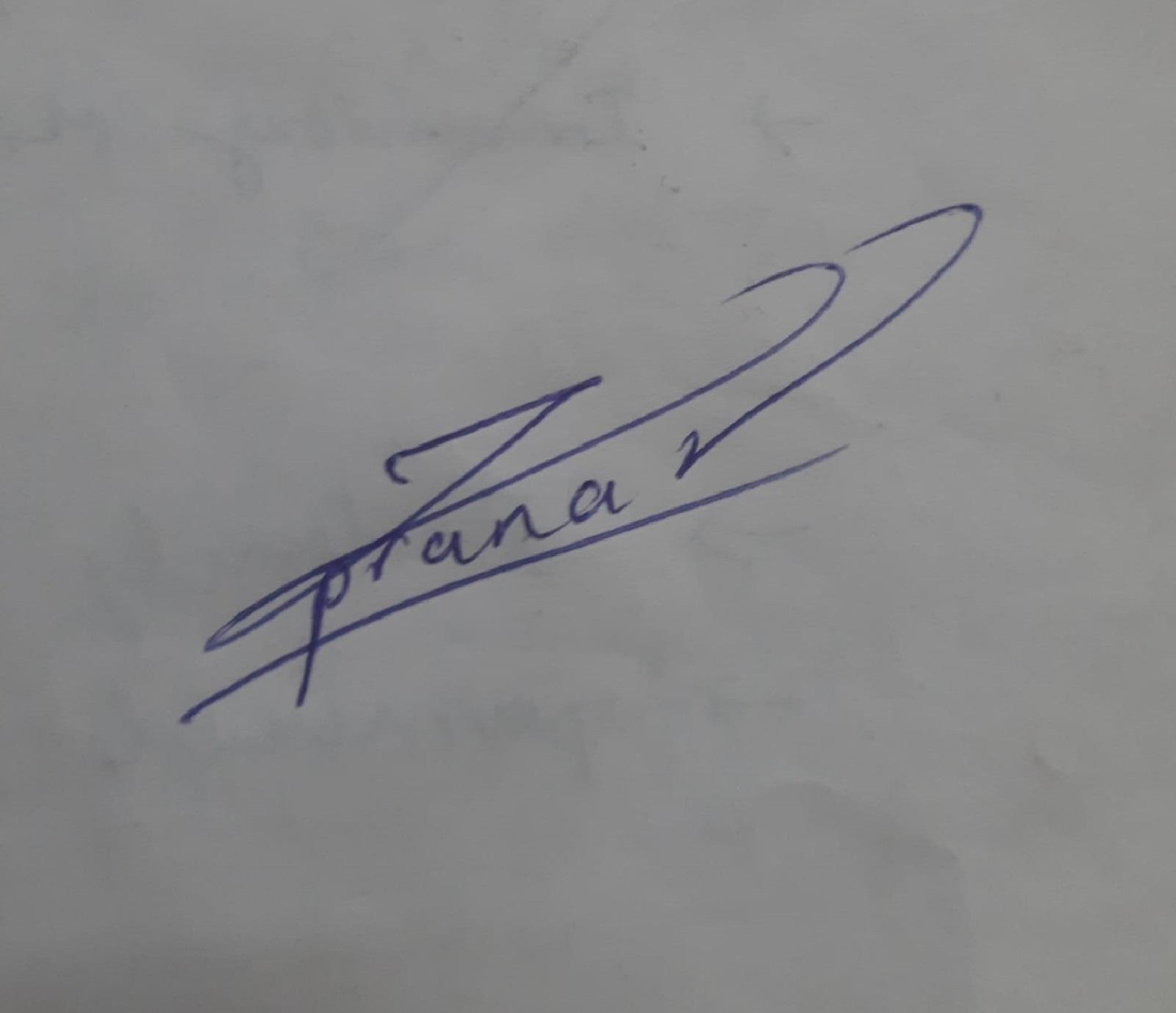
* + Hall tickets have to be returned to the invigilator before leaving the exam hall. No paper can be taken out of the exam hall.
  + Press the SUBMIT button on the computer after you have completed the exam.

**IMPORTANT:**

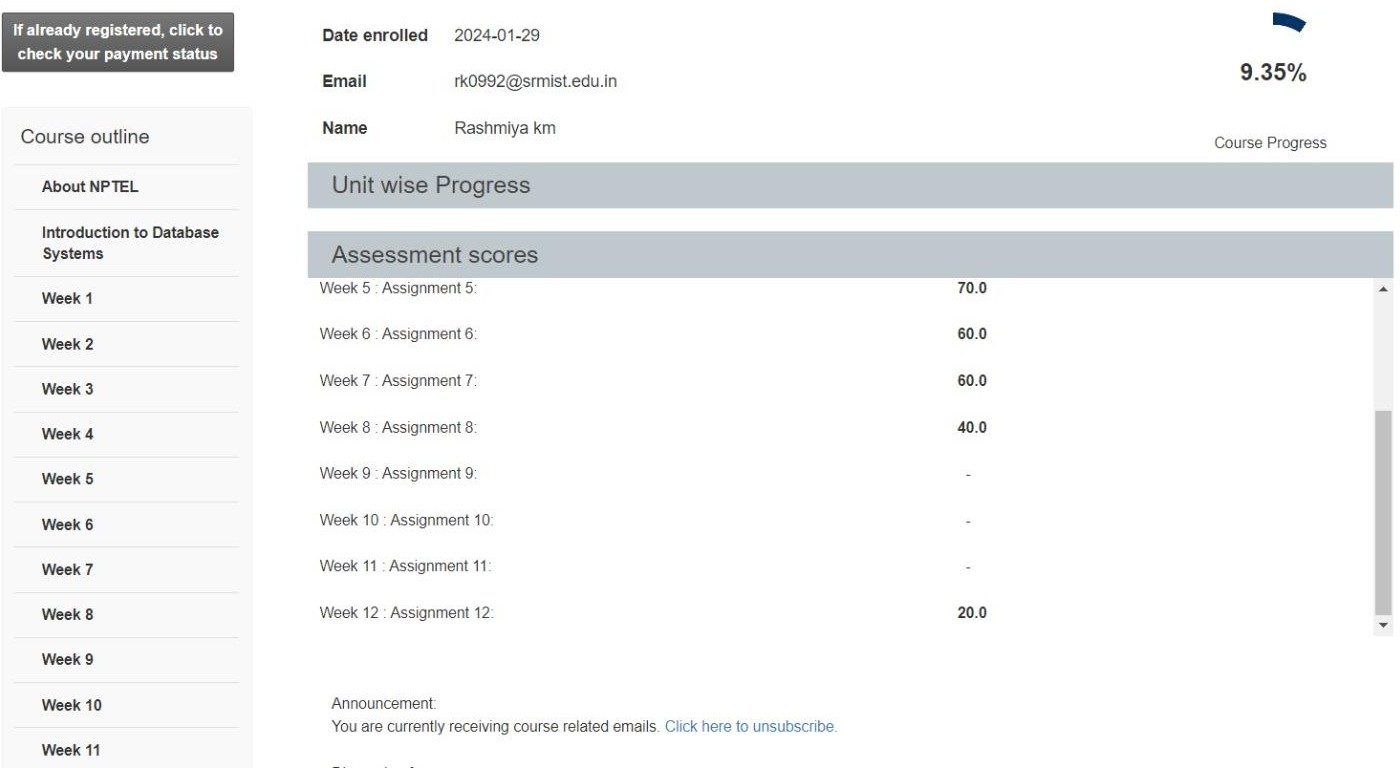
* + A basic code of conduct during the exam should be followed, failing which, NPTEL reserves the right to take appropriate action.
  + In case the exam is delayed due to any unforeseen circumstances, NPTEL will decide on the appropriate course of action as it deems fit.

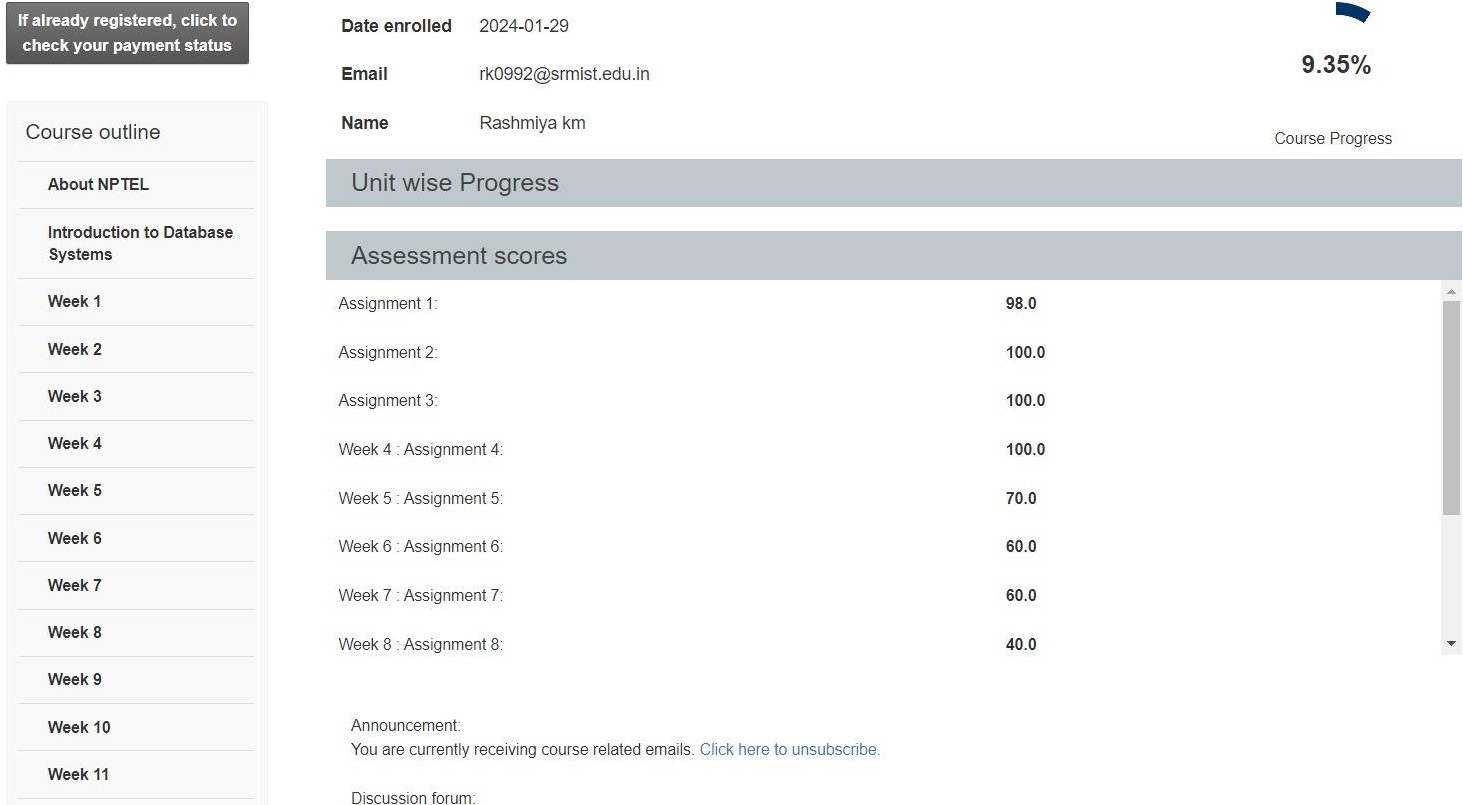
**AT THE EXAM CENTRE, IF YOU ENCOUNTER ANY ISSUES WITH RESPECT TO THE COMPUTER OR EXAM OFFICIALS, KINDLY CONTACT THE NPTEL EXAM REPRESENTATIVE, WHO WILL BE AVAILABLE AT THE CENTRE.**

I HEREBY ACKNOWLEDGE THAT I HAVE READ, UNDERSTOOD AND AGREE TO FOLLOW THE ABOVE MENTIONED INSTRUCTIONS.



**Signature of the Candidate**





NPTEL

(Signature of Candidate)

|  |  |  |
| --- | --- | --- |
| 6ÏTEROO SESSIO  (6) | National Programme on Technology Enhanced Learning |  |
| Hall Ticket For | 2024 Apr: CS55 Introduction to Database Systems - Online |

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Candidate Name | Rashmiya km | | | | | | | | | |  |
| Roll No | NOC24CS55S653405019 | | | | | Seating Number | | | 53405019 | |
| Date of Birth | 30-03-2005 | | | | | | | | | |
| PwD Status | No | Compensatory Time Required | | | N.A | | Scribe Required | | | N.A |
| Exam Date | Sunday, 21 April, 2024 | | | | | | | | | |
| Reporting Time | 01:00 pm | | | Gate Closure | | | | 02:30 pm | | |
| Exam Timing | 02:00 pm | | | Shift | | | | AN | | |
| Test Centre Name | Sri Sai Ram Engineering College | | | | | | | | | |  |
| Test Centre Address | Sai Leo Nagar, Sairam Rd, West Tambaram, , Chennai, Tamil Nadu, India - 600044 | | | | | | | | | |
| **NPTEL Coordinator** | | |  | | | | | | | | |

|  |
| --- |
| **NPTEL EXAM - 21 APRIL, 2024**  **General instructions for candidates - AN**  (All timings mentioned here are in IST) |
| **The total duration of the examination is 180 minutes.**  **Candidates will be permitted to leave the examination hall only after 03:30 pm, on a need basis.** |
| **Hall ticket and Entry:**   1. The Hall Ticket must be presented for verification along with one original photo identification (not photocopy or scanned copy). Examples of acceptable photo identification documents are School ID, College ID, Employee ID, Driving License, Passport, PAN card, Voter ID, and Aadhaar-ID. A printed copy of the hall ticket and original photo ID card should be brought to the exam centre. Hall ticket and ID card copies on the phone will not be permitted. 2. This Hall Ticket is valid only if the candidate’s photograph and signature images are legible. To ensure this, print the Hall Ticket on A4-sized paper using a laser printer, preferably a color photo printer. 3. **TIMELINE:** 1:00 pm - Report to the examination venue | 1:40 pm – Candidates will be permitted to occupy their allotted seats| 1:50 pm – Candidates can login and start reading instructions prior to the examination | 2:00 pm - Exam starts |   2:30 pm - Gate closes, candidates will not be allowed after this time | 3:30 pm Submit button will be enabled | 5:00 pm exam ends.   1. Candidates will be permitted to appear for the examination ONLY after their credentials are verified by center officials.   **P.T.O.** |

1. Candidates are advised to locate the examination center at least a day prior to the examination, so that they can reach the center on time for the examination.

**STATIONERY REQUIREMENTS:**

* + A4 sheets will be provided to candidates for rough work. Candidates have to write their name and registration number on the A4 Sheets before they start using it. The A4 sheets must be returned to the invigilator at the end of the examination.
  + On-screen calculator will be available during the exam. Candidates are advised to familiarize themselves with this virtual Scientific calculator well ahead of the exam.

Link: [https://www.tcsion.com/OnlineAssessment/ScientificCalculator/Calculator.html](http://www.tcsion.com/OnlineAssessment/ScientificCalculator/Calculator.html)

* + You should bring your own pen/pencil; it will NOT be given at the examination centre.

**DRESS CODE:**

* + Candidates are expected to come in professional attire to write the exams.
  + Candidates wearing SHORTS will NOT be permitted inside the exam hall.

**PERMITTED:**

* + You may bring vehicle keys inside the exam hall.
  + You are advised to carry your own drinking water in a transparent bottle.
  + Candidates are allowed to bring sanitizer in a small transparent bottle.

**NOT PERMITTED:**

* + Watches, wallets, mobile phones, Bluetooth devices, microphones, pagers, health bands or any other electronic gadgets, any printed/blank/handwritten paper, log tables, writing pads, scales, geometry/pencil-boxes, pouches, calculators, pen drives, electronic pens, handbags, goggles, electronic vehicle keys or similar such items are NOT allowed inside the examination centre. There may not be any facility for the safekeeping of these devices outside the examination hall; it will be prudent not to bring valuables to the examination center. Candidates will not be permitted to carry any food items in the exam centre. We suggest that you bring a bag to keep routine belongings outside the exam hall. Neither NPTEL nor the exam provider takes responsibility for the bag and the belongings. You may keep it outside at your own risk.

**MANDATORY :**

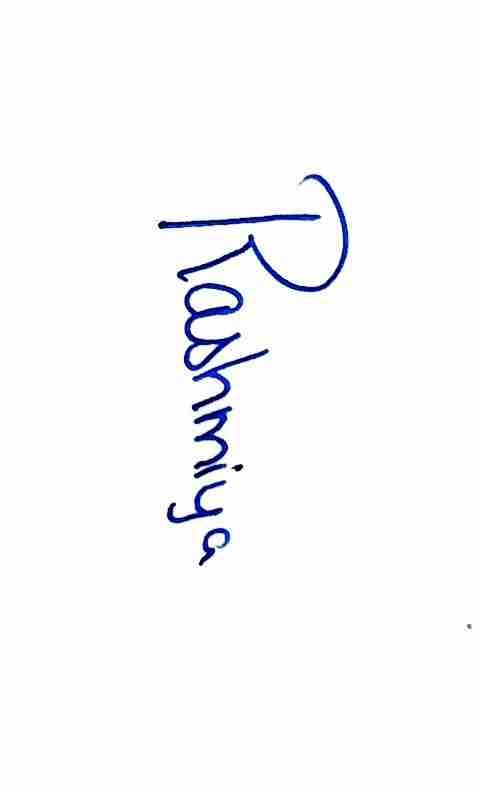
* + Hall tickets have to be returned to the invigilator before leaving the exam hall. No paper can be taken out of the exam hall.
  + Press the SUBMIT button on the computer after you have completed the exam.

**IMPORTANT:**

* + A basic code of conduct during the exam should be followed, failing which, NPTEL reserves the right to take appropriate action.
  + In case the exam is delayed due to any unforeseen circumstances, NPTEL will decide on the appropriate course of action as it deems fit.

**AT THE EXAM CENTRE, IF YOU ENCOUNTER ANY ISSUES WITH RESPECT TO THE COMPUTER OR EXAM OFFICIALS, KINDLY CONTACT THE NPTEL EXAM REPRESENTATIVE, WHO WILL BE AVAILABLE AT THE CENTRE.**

I HEREBY ACKNOWLEDGE THAT I HAVE READ, UNDERSTOOD AND AGREE TO FOLLOW THE ABOVE MENTIONED INSTRUCTIONS.



**Signature of the Candidate**