

VISVESVARAYA TECHNOLOGICAL UNIVERSITY
“JNANA SANGAMA”, BELAGAVI - 590 018



A MINI PROJECT REPORT
on
“ ARMS AND AMMUNITION SYSTEM ”

Submitted by

Krishna G Kamath	4SF18IS044
Mayur Pai B H	4SF18IS053

In partial fulfillment of the requirements for the V semester

DBMS LABORATORY WITH MINI PROJECT
of
BACHELOR OF ENGINEERING
in
INFORMATION SCIENCE & ENGINEERING

Under the Guidance of

Mr. Rithesh Pakkala P.

Assistant Professor

Department of ISE

at



SAHYADRI

College of Engineering & Management

Adyar, Mangaluru - 575 007

2020 - 21

SAHYADRI
College of Engineering & Management
Adyar, Mangaluru - 575 007

Department of Information Science & Engineering



CERTIFICATE

This is to certify that the **Mini Project** entitled “**Arms and Ammunition Management System**” has been carried out by **Krishna G Kamath (4SF18IS044)** and **Mayur Pai B H (4SF18IS053)**, the bonafide students of Sahyadri College of Engineering & Management in partial fulfillment of the requirements for the V semester **DBMS Laboratory with Mini Project (18CSL58)** of **Bachelor of Engineering in Information Science & Engineering** of Visvesvaraya Technological University, Belagavi during the year 2020 - 21. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report deposited in the departmental library. The mini project report has been approved as it satisfies the academic requirements in respect of mini project work.

Mr. Rithesh Pakkala P.
Assistant Professor
Dept. of ISE, SCEM

Dr. Shamanth Rai
HOD & Associate Professor
Dept. of ISE, SCEM

External Practical Examination:

Examiner's Name

Signature with Date

1.

.....

2.

.....

SAHYADRI
College of Engineering & Management
Adyar, Mangaluru - 575 007

Department of Information Science & Engineering



DECLARATION

We hereby declare that the entire work embodied in this Mini Project Report titled **“Arms and Ammunition Management System”** has been carried out by us at Sahyadri College of Engineering and Management, Mangaluru under the supervision of **Mr. Rithesh Pakkala P.** as the part of the V semester **DBMS Laboratory with Mini Project (18CSL58)** of **Bachelor of Engineering in Information Science & Engineering**. This report has not been submitted to this or any other University.

Krishna G Kamath (4SF18IS044)

Mayur Pai B H (4SF18IS053)

SCEM, Mangaluru

Abstract

Arms and Ammunition Management System is designed in order to store and validate the information of weapons and equipments. The database is updated at regular intervals of time with the accurate count of firearms in stock. The arms and ammo that are in deficit are noted down in order for purchase. The database is an end product that has the potential to serve the defence sector of the nation. Its requirements are to provide the basic information maintenance function of weapons, manufacturers and ammos so that users can use the function to add, delete, and modify the basic information of weapons. Arms and Ammunition Management System is very convenient for managing input, output, and searching the weapons so as to make the work of users efficient and effective. In the aspect of software, the Arms and Ammunition Management System uses Python language and SQLAlchemy as the background database. Various configurations in computer including input and output capacity, internal memory and external memory capacity are met the requirements of users.

Acknowledgement

It is with great satisfaction and euphoria that we are submitting the Mini Project Report on “**Arms and Ammunition Management System**”. We have completed it as a part of the V semester **DBMS Laboratory with Mini Project (18CSL58)** of **Bachelor of Engineering in Information Science & Engineering** of Visvesvaraya Technological University, Belagavi.

We are profoundly indebted to our guide, **Mr. Rithesh Pakkala P.**, Assistant Professor, Department of Information Science & Engineering for innumerable acts of timely advice, encouragement and We sincerely express our gratitude.

We express our sincere gratitude to **Dr. Shamanth Rai**, Head & Associate Professor, Department of Information Science & Engineering for his invaluable support and guidance.

We sincerely thank **Dr. Rajesha S**, Principal, Sahyadri College of Engineering & Management and **Dr. D. L. Prabhakara**, Director, Sahyadri Educational Institutions, who have always been a great source of inspiration.

Finally, yet importantly, We express our heartfelt thanks to our family & friends for their wishes and encouragement throughout the work.

Krishna G Kamath

4SF18IS044

V Sem, B.E., ISE

SCEM, Mangaluru

Mayur Pai B H

4SF18IS053

V Sem, B.E., ISE

SCEM, Mangaluru

Table of Contents

Abstract	i
Acknowledgement	ii
Table of Contents	iv
List of Figures	v
1 Introduction	1
1.1 Database Management System	1
1.2 Structured Query Language	2
1.3 Stored Procedure	2
1.4 Trigger	3
1.5 Normalisation	3
1.6 Application	3
2 Hardware and Software Details	4
2.1 Hardware Details	4
2.2 Software Details	4
3 System Design	5
3.1 Entity Relation Diagram(ERD)	5
3.1.1 Entity	6
3.1.2 Weak Entity	6
3.1.3 Attribute	6
3.1.4 Multivalued Attribute	6
3.1.5 Derived Attribute	6
3.1.6 Relationship	7
3.1.7 Recursive Relationship	7
3.1.8 Cardinality Ratio	7
3.2 Mapping From ER Diagram to Schema Diagram	8
3.3 Schema Diagram	11

4	Implementation	13
4.1	Table Structure	13
4.2	Codes Used For Modules:	20
5	Results and Discussion	23
6	Conclusion and Future work	27
	References	28

List of Figures

3.1	Components of ER Diagram	5
3.2	ER Diagram of Arms and Ammunition Management System	7
3.3	Mapping of Regular Entities	9
3.4	Mapping of binary M:N Relation Types	10
3.5	Mapping of Weak Entity	11
3.6	Schema Diagram of Arms and Ammunition Management System	12
5.1	Login Page	23
5.2	Registration Page	23
5.3	Delete Weapons	24
5.4	Update Weapon Details	24
5.5	Search Weapon Details	25
5.6	Trigger for Modification of Weapons	25
5.7	Mail Generated	26

Chapter 1

Introduction

Arms and Ammunition Management System is designed in order to store and validate the information of weapons and equipments. The database is updated at regular intervals of time with the accurate count of firearms in stock. The arms and ammos that are in deficit are noted down in order for purchase. The database is an end product that has the potential to serve the defence sector of the nation.

1.1 Database Management System

With the widespread use of computer technology and network technology, the development of database technology has become an important part of advanced information technology. The core of the Arms and Ammunition Management System is how to use and operate database, so the database design is critical. This system uses the SQLAlchemy which is the Python SQL toolkit and Object Relational Mapper that gives application developers the full power and flexibility of SQL. It has quickly become one of the most widely used object-relational mapping tools in the Python community, alongside Django's ORM, which is easy to use, strong function and suitable for all kinds of large, medium and small, microcomputer environment. It can realize data sharing and the facilities don't need to have the powerful data storage and processing capabilities so that to reduce the hardware cost. Many enterprises have their own database, and store a large number of key data in it, which shows the importance of the database.

Every table in the database is broken up into smaller entities called fields. The fields in the Country table consist of Cname, Clocation. A field is a column in a table that is designed to maintain specific information about every record in the table. A record,

also called a row, is each individual entry that exists in a table. A record is a horizontal entity in a table. A column is a vertical entity in a table that contains all information associated with a specific field in a table. In addition to tables, a database can also contain other objects including views, stored procedures, indexes and constraints, along with a transaction log.

1.2 Structured Query Language

SQL is a domain specific language used in programming and designed for managing data held in a database management system. SQL consists of a data definition language, data manipulation language and data control language. The scope of SQL includes data insert, query update and delete, schema creation and modification and data access control. As a database server, it is a software product with the primary function of storing and retrieving data as requested by other software applications which may run either on the same computer or on another computer across a network. The main mode of retrieving data from a SQL Server database is querying for it. The query declaratively specifies what is to be retrieved. It is processed by the query processor, which figures out the sequence of steps that will be necessary to retrieve the requested data. The sequence of actions necessary to execute a query is called a query plan. There might be multiple ways to process the same query. Stored procedures can accept values sent by the client as input parameters, and send back results as output parameters. They can call defined functions, and other stored procedures, including the same stored procedure.

1.3 Stored Procedure

A stored procedure is a prepared SQL code that you can save, so the code can be reused over and over again. So if you have an SQL query that you write over and over again, save it as a stored procedure, and then just call it to execute it. We can also pass parameters to a stored procedure, so that the stored procedure can act based on the parameter value(s) that is passed. A stored procedure is nothing more than prepared SQL code that you save so you can reuse the code over and over again. So if you think about a query that you write over and over again, instead of having to write that query each time you would save it as a stored procedure and then just call the stored procedure to execute the SQL code that you saved as part of the stored procedure. In addition to running the same SQL code

over and over again you also have the ability to pass parameters to the stored procedure, so depending on what the need is the stored procedure can act accordingly based on the parameter values that were passed.

1.4 Trigger

A SQL trigger is a set of SQL statements stored in the database catalog. A SQL trigger is executed or fired whenever an event associated with a table occurs e.g., insert, update or delete. A SQL trigger is a special type of stored procedure. It is special because it is not called directly like a stored procedure. The main difference between a trigger and a stored procedure is that a trigger is called automatically when a data modification event is made against a table whereas a stored procedure must be called explicitly.

1.5 Normalisation

Database Normalization is a technique of organizing the data in the database. Normalization is a systematic approach of decomposing tables to eliminate data redundancy (repetition) and undesirable characteristics like Insertion, Update and Deletion Anomalies. It is a multi-step process that puts data into tabular form, removing duplicated data from the relation tables.

Normalization is used for mainly two purposes:

- i. Eliminating redundant (useless) data.
- ii. Ensuring data dependencies make sense i.e data is logically stored.

1.6 Application

This application is designed in order to store and validate the information of weapons and equipments. The database is updated at regular intervals of time with the accurate count of firearms in stock. The arms and ammos that are in deficit are noted down in order for purchase. The database is an end product that has the potential to serve the defence sector of the nation.

Chapter 2

Hardware and Software Details

2.1 Hardware Details

- Processor : Any Processor more than 500 MHz
- RAM : 2GB
- Hard Disk : 500GB
- Input Device : Standard Keyboard and Mouse
- Output Device : Monitor

2.2 Software Details

- Database : MySQL/MariaDB
- Programming Language : Python
- IDE : Visual Studio Code
- Operating System : Microsoft Windows 10 Pro

Chapter 3

System Design

3.1 Entity Relation Diagram(ERD)

An entity relationship diagram(ERD), also known as entity relationship model, is a graphical representation of an information system that depicts the relationship among people, object, places, concepts or events within that system. An ERD is a data modeling technique that can help to define business processes and be used as a foundation for a relational database.

There are three basic elements in an ER Diagram: entity, attribute, relationship. There are more elements which are based on the main elements. They are weak entity, multivalued attribute, derived attribute, weak relationship, and recursive relationship. Cardinality and ordinality are two other notations used in ER diagrams to further define relationships.

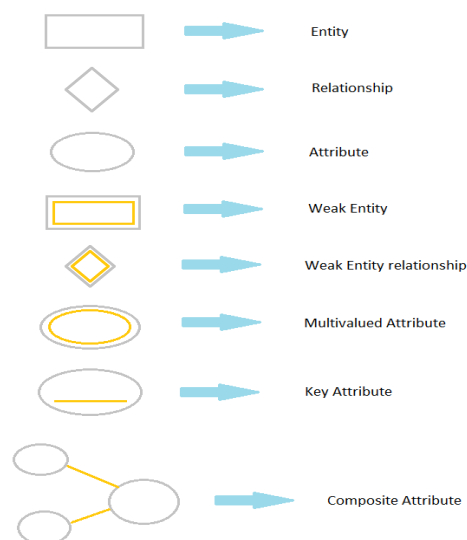


Figure 3.1: Components of ER Diagram

3.1.1 Entity

An entity can be a person, place, event, or object that is relevant to a given system. For example, a school system may include students, teachers, major courses, subjects, fees and other items. Entities are represented in ER diagrams by a rectangle and named using singular nouns.

3.1.2 Weak Entity

A weak entity is an entity that depends on the existence of another entity. In more technical terms it can be defined as an entity that cannot be identified by its own attributes. It uses a foreign key combined with its attributed to form the primary key. An entity like order item is a good example for this. The order item will be meaningless without an order so it depends on the existence of the order.

3.1.3 Attribute

An attribute is a property, trait, or characteristic of an entity, relationship, or another attribute. For example, the attribute Inventory Item Name is an attribute of the entity Inventory Item. An entity can have as many attributes as necessary. Meanwhile, attributes can also have their own specific attributes. For example, the attribute “customer address” can have the attributes number, street, city, and state. These are called composite attributes. Note that some top level ER diagrams do not show attributes for the sake of simplicity. In those that do, however, attributes are represented by oval shapes.

3.1.4 Multivalued Attribute

If an attribute can have more than one value it is called a multivalued attribute. It is important to note that this is different from an attribute having its own attributes. For example, a teacher entity can have multiple subject values.

3.1.5 Derived Attribute

An attribute based on another attribute. This is found rarely in ER diagrams. For example, for a circle, the area can be derived from the radius.

3.1.6 Relationship

A relationship describes how entities interact. Relationships are represented by diamond shapes and are labelled using verbs.

3.1.7 Recursive Relationship

If the same entity participates more than once in a relationship it is known as a recursive relationship. In the below example an employee can be a supervisor and be supervised, so there is a recursive relationship.

3.1.8 Cardinality Ratio

There are three types of Cardinality Ratios:

- 1:1
- 1:N
- M:N

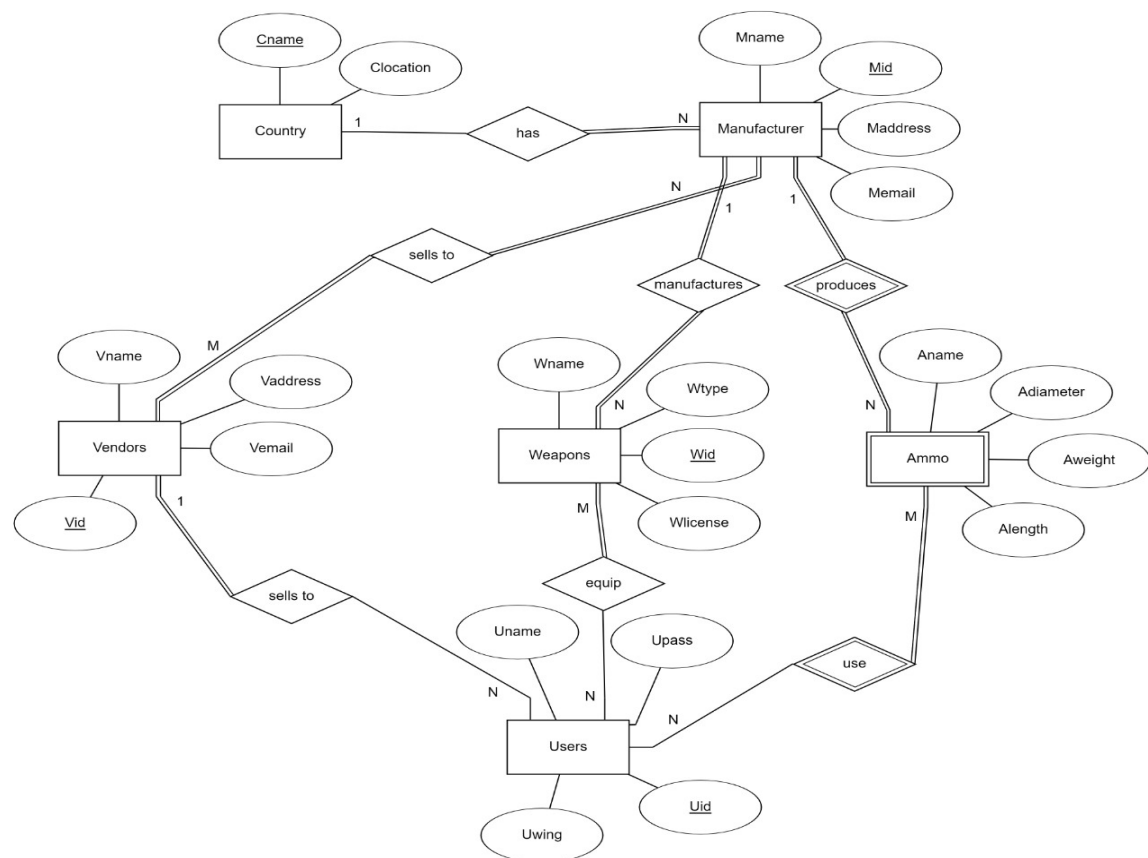


Figure 3.2: ER Diagram of Arms and Ammunition Management System

3.2 Mapping From ER Diagram to Schema Diagram

1.Mapping of Regular Entities:This step involves mapping all the regular entity types to tabular format by identifying their primary keys.

2.Mapping of 1:1 Relation:In this step foreign keys are assigned using foreign key approach.The primary key of the participating relation R or S is added as primary key to second entity types by looking at the participating constraints.

3.Mapping of 1:N Relation:Foreign key approach is used to add one sided primary key to the n sided entity at foreign key.

4.Mapping of M:N Relation:Here we use the cross reference approach where the relationship is converted to a new relation within attributes on primary keys of both participating relation.

5.Mapping of Weak Entity:When mapping weak entity types along with other attributes the partial key and primary key of parent entity together will form their primary key of the new relation.

6.Mapping of N-ary Relation:For mapping N array relationship we create a new relation with a relationship name in its attribute and primary keys of all participating entity types.

7.Mapping of Multivalued Relation:For multivalued attributes a separate relation has to be created along with primary key of parent relation.

In our database we have the following mappings:

Step – 1 : Mapping of Regular Entities.

From the ER diagram we identify all the strong entities E and create a relation R that includes all it's simple attributes and primary keys.

The following are the strong entities from our schema diagram :

- 1.COUNTRY(Cname,Clocation)
- 2.MANUFACTURER(Mid,Mname,Maddress,Memail)
- 3.VENDORS(Vid,Vname,Vaddress,Vemail)
- 4.WEAPONS(Wid,Wname,Wtype,Wlicense)
- 5.USERS(Uid,Uname,Uwing)

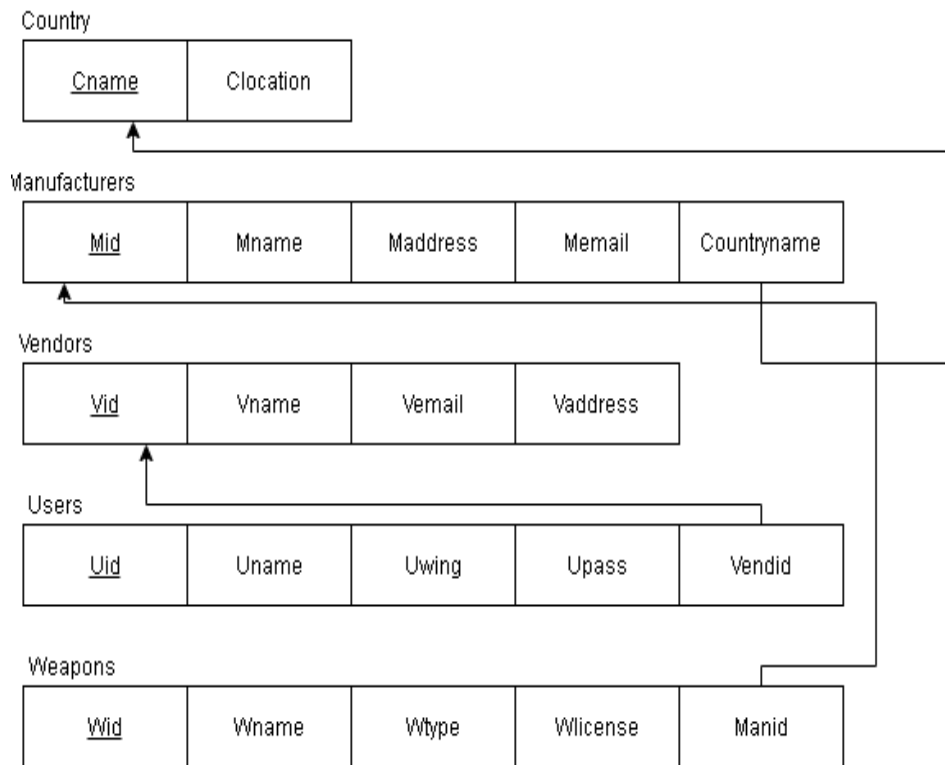


Figure 3.3: Mapping of Regular Entities

Step – 2 : Mapping of binary 1:1 Relation Types.

In relational database design, a one-to-one (1:1) relationship exists when zero or one instance of entity A can be associated with zero or one instance of entity B, and zero or one instance of entity B can be associated with zero or one instance of entity A.

Unfortunately, we don't have any 1:1 relation existing within our database design.

Step – 3 : Mapping of binary 1:N Relation Types.

The COUNTRY and the MANUFACTURER entities are participating in the 1:N relation type. Since MANUFACTURER is on the nth side of the relation we include the primary key of COUNTRY entity as the Foreign key in MANUFACTURER entity.

The MANUFACTURER and the WEAPONS entities are participating in the 1:N relation type. Since WEAPONS is on the nth side of the relation we include the primary key of MANUFACTURER entity as the Foreign key in WEAPONS entity.

The VENDORS and the USERS entities are participating in the 1:N relation type. Since USERS is on the nth side of the relation we include the primary key of VENDORS entity as the Foreign key in USERS entity.

The MANUFACTURER and the AMMOS entities are participating in the 1:N relation type. Since AMMOS is on the nth side of the relation we include the primary key of

MANUFACTURER entity as the Foreign key in AMMOS entity.

Step – 4 : Mapping of binary M:N Relation Types.

The relationship between the VENDORS and the MANUFACTURER is M:N .So we create a new relation SELLS TO which includes the primary key of VENDORS AND MANUFACTURER entity. The combination of the two primary keys will form the primary key of the SELLS TO relation.

The relationship between the WEAPONS and the USERS is M:N .So we create a new relation EQUIP which includes the primary key of WEAPONS AND USERS entity. The combination of the two primary keys will form the primary key of the EQUIP relation.

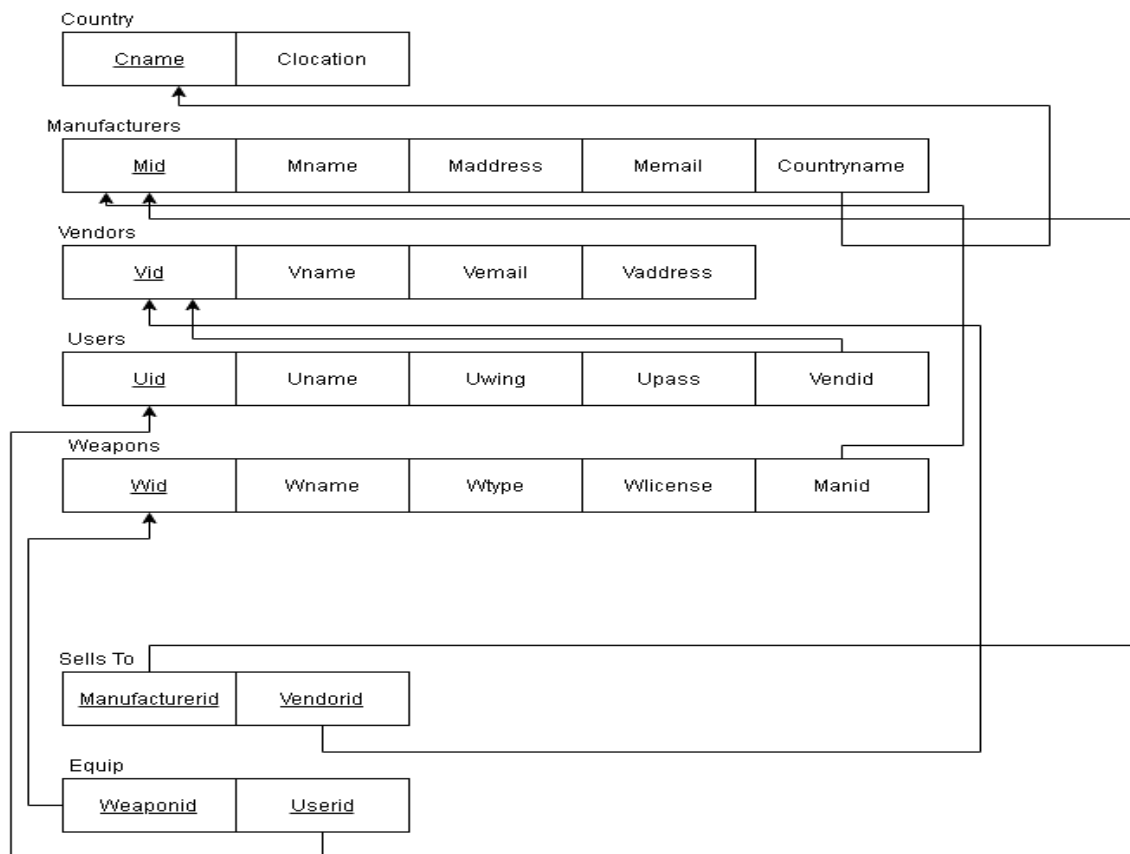


Figure 3.4: Mapping of binary M:N Relation Types

Step – 5 : Mapping of Weak Entity.

From the ER diagram we identify all the weak entities E and create a relation R that includes all it's simple attributes and partial keys.

The following are the weak entities from our schema diagram :

1.AMMOS(Aname,Adiameter,Aweight,Alength)

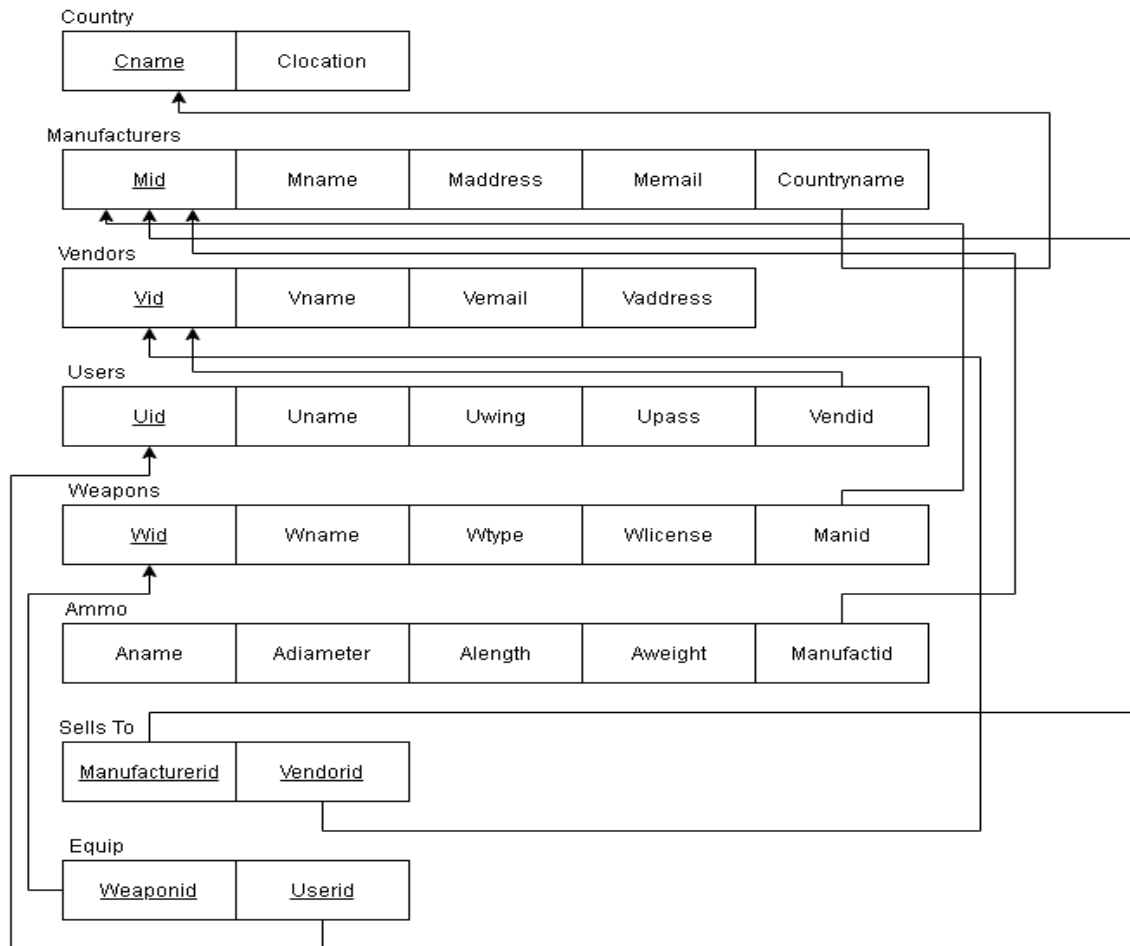


Figure 3.5: Mapping of Weak Entity

3.3 Schema Diagram

A Schema is a pictorial representation of the relationship between the database tables in the database that is created. The database schema of a database system is its structure described in a formal language supported by the database management system (DBMS). The term "schema" refers to the organization of data as a blueprint of how the database is constructed (divided into database tables in the case of relational databases).

The formal definition of a database schema is a set of formulas (sentences) called integrity constraints imposed on a database. These integrity constraints ensure compatibility be-

tween parts of the schema. All constraints are expressible in the same language. A database can be considered a structure in realization of the database language. The states of a created conceptual schema are transformed into an explicit mapping, the database schema. This describes how real-world entities are modelled in the database.

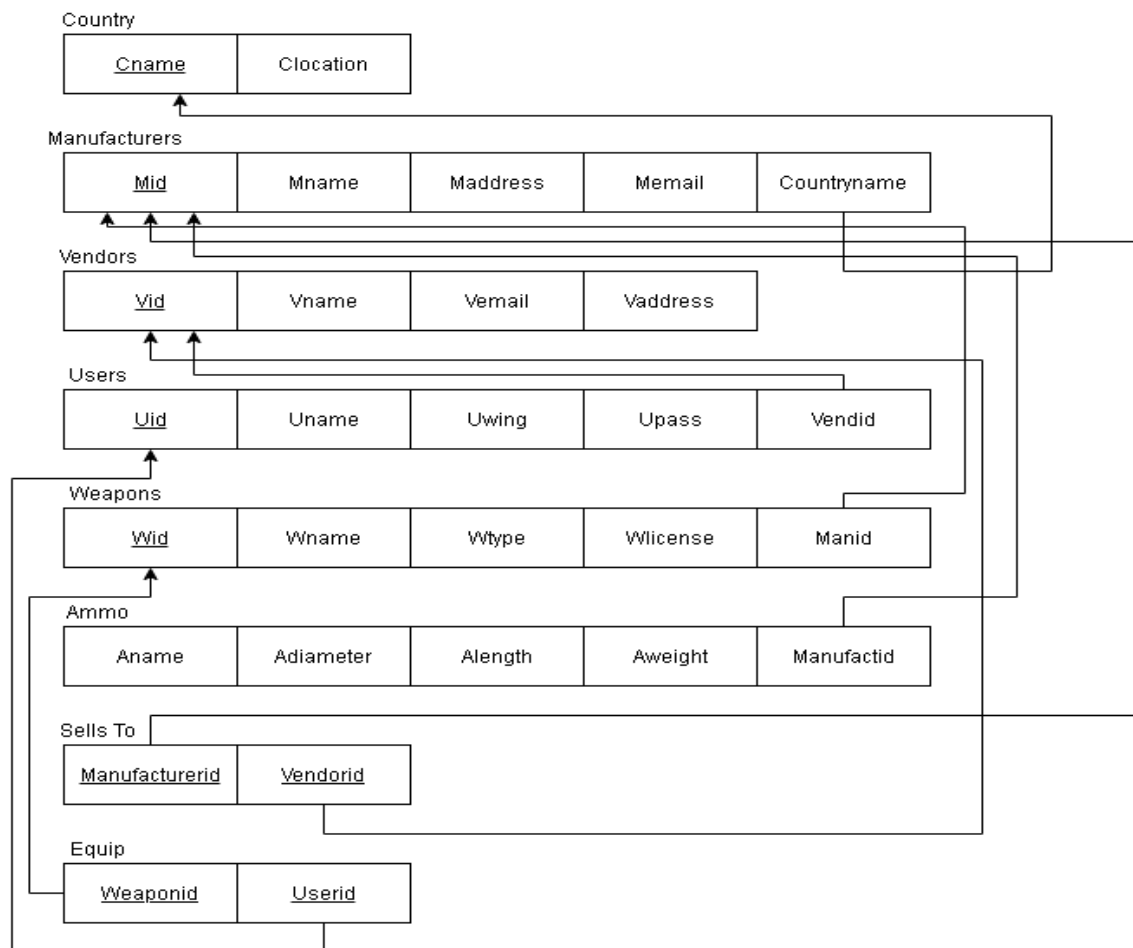


Figure 3.6: Schema Diagram of Arms and Ammunition Management System

Chapter 4

Implementation

4.1 Table Structure








AMMOS

```
CREATE TABLE 'ammos' (  
  'Aname' varchar(50) NOT NULL,  
  'Adiameter' varchar(20) NOT NULL,  
  'Alength' varchar(20) NOT NULL,  
  'Aweight' varchar(20) NOT NULL,  
  'Manufactid' int(10) NOT NULL  
) ENGINE=InnoDB DEFAULT CHARSET=latin1;  
  
ALTER TABLE 'ammos'  
  
ADD CONSTRAINT 'Manufactid' FOREIGN KEY  
( 'Manufactid' ) REFERENCES 'manufacturers' ( 'Mid' ) ON DELETE CASCADE ON  
UPDATE CASCADE;
```

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
<input type="checkbox"/> 1	Aname	varchar(50)	latin1_swedish_ci		No	None			Change Drop More
<input type="checkbox"/> 2	Adiameter	varchar(20)	latin1_swedish_ci		No	None			Change Drop More
<input type="checkbox"/> 3	Alength	varchar(20)	latin1_swedish_ci		No	None			Change Drop More
<input type="checkbox"/> 4	Aweight	varchar(20)	latin1_swedish_ci		No	None			Change Drop More
<input type="checkbox"/> 5	Manufactid	int(10)			No	None			Change Drop More









COUNTRY

```
CREATE TABLE 'country' (
  'Cname' varchar(50) NOT NULL,
  'Clocation' varchar(50) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
ALTER TABLE 'country'
ADD PRIMARY KEY ('Cname');
```

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
<input type="checkbox"/> 1	Cname 	varchar(50)	latin1_swedish_ci		No	None			 Change  Drop  More
<input type="checkbox"/> 2	Clocation	varchar(50)	latin1_swedish_ci		No	None			 Change  Drop  More

EQUIP

```
CREATE TABLE 'equip' (
  'Weaponid' int(10) NOT NULL,
  'Userid' int(10) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
ALTER TABLE 'equip'
ADD CONSTRAINT 'Userid' FOREIGN KEY ('Userid') REFERENCES 'users'
('Uid') ON DELETE CASCADE ON UPDATE CASCADE,
ADD CONSTRAINT 'Weaponid' FOREIGN KEY ('Weaponid') REFERENCES
'weapons' ('Wid') ON DELETE CASCADE ON UPDATE CASCADE;
```


















#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
<input type="checkbox"/> 1	Weaponid 	int(10)			No	None			 Change  Drop  More
<input type="checkbox"/> 2	Userid 	int(10)			No	None			 Change  Drop  More

MANUFACTURER

```

CREATE TABLE 'manufacturers' (
'Mid' int(10) NOT NULL,
'Mname' varchar(50) NOT NULL,
'Maddress' varchar(50) NOT NULL,
'Memail' varchar(50) NOT NULL,
'Countryname' varchar(50) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
ALTER TABLE 'manufacturers'
ADD PRIMARY KEY ('Mid'),
ADD KEY 'Countryname' ('Countryname');
ALTER TABLE 'manufacturers'
ADD CONSTRAINT 'Countryname' FOREIGN KEY ('Countryname') REFERENCES
'country' ('Cname') ON DELETE CASCADE ON UPDATE CASCADE;

```









#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
<input type="checkbox"/> 1	Mid 	int(10)			No	None			 Change  Drop  More
<input type="checkbox"/> 2	Mname	varchar(50)	latin1_swedish_ci		No	None			 Change  Drop  More
<input type="checkbox"/> 3	Maddress	varchar(50)	latin1_swedish_ci		No	None			 Change  Drop  More
<input type="checkbox"/> 4	Memail	varchar(50)	latin1_swedish_ci		No	None			 Change  Drop  More
<input type="checkbox"/> 5	Countryname 	varchar(50)	latin1_swedish_ci		No	None			 Change  Drop  More

SELLS TO

```

CREATE TABLE 'sellsto' (
  'Manufactureid' int(10) NOT NULL,
  'Vendorid' int(10) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
ALTER TABLE 'sellsto'
ADD KEY 'Manufactureid' ('Manufactureid'),
ADD KEY 'Vendorid' ('Vendorid');
ALTER TABLE 'sellsto'
ADD CONSTRAINT 'Manufactureid' FOREIGN KEY ('Manufactureid')
REFERENCES 'manufacturers' ('Mid') ON DELETE CASCADE ON UPDATE
CASCADE,
ADD CONSTRAINT 'Vendorid' FOREIGN KEY ('Vendorid') REFERENCES
'vendors' ('Vid') ON DELETE CASCADE ON UPDATE CASCADE;

```


















#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
<input type="checkbox"/> 1	Manufactureid 	int(10)			No	None			 Change  Drop  More
<input type="checkbox"/> 2	Vendorid 	int(10)			No	None			 Change  Drop  More

USERS

```

CREATE TABLE 'users' (
'Uid' int(10) NOT NULL,
'Uname' varchar(100) NOT NULL,
'Uwing' varchar(50) NOT NULL,
'Upass' varchar(100) NOT NULL,
'Vendid' int(10) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
ALTER TABLE 'users'
ADD PRIMARY KEY ('Uid'),
ADD KEY 'Vendid' ('Vendid');
ALTER TABLE 'users'
ADD CONSTRAINT 'Vendid' FOREIGN KEY ('Vendid') REFERENCES 'vendors'
('Vid') ON DELETE CASCADE ON UPDATE CASCADE;

```














#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
<input type="checkbox"/> 1	Uid 	int(10)			No	None			 Change  Drop  More
<input type="checkbox"/> 2	Uname	varchar(100)	latin1_swedish_ci		No	None			 Change  Drop  More
<input type="checkbox"/> 3	Uwing	varchar(50)	latin1_swedish_ci		No	None			 Change  Drop  More
<input type="checkbox"/> 4	Upass	varchar(100)	latin1_swedish_ci		No	None			 Change  Drop  More
<input type="checkbox"/> 5	Vendid 	int(10)			No	None			 Change  Drop  More

VENDORS

```

CREATE TABLE 'vendors' (
'Vid' int(10) NOT NULL,
'Vname' varchar(20) NOT NULL,
'Vemail' varchar(50) NOT NULL,
'Vaddress' varchar(50) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
ALTER TABLE 'vendors'
ADD PRIMARY KEY ('Vid');

```

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
<input type="checkbox"/> 1	Vid 	int(10)			No	None		AUTO_INCREMENT	 Change  Drop  More
<input type="checkbox"/> 2	Vname	varchar(20)	latin1_swedish_ci		No	None			 Change  Drop  More
<input type="checkbox"/> 3	Vemail	varchar(50)	latin1_swedish_ci		No	None			 Change  Drop  More
<input type="checkbox"/> 4	Vaddress	varchar(50)	latin1_swedish_ci		No	None			 Change  Drop  More

WEAPONS

```

CREATE TABLE 'weapons' (
'Wid' int(10) NOT NULL,
'Wname' varchar(40) NOT NULL,
'Wtype' varchar(40) NOT NULL,
'Wlicense' varchar(20) NOT NULL,
'Manid' int(10) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
ALTER TABLE 'weapons'
ADD PRIMARY KEY ('Wid'),
ADD KEY 'Manid' ('Manid');
ALTER TABLE 'weapons'
ADD CONSTRAINT 'Manid' FOREIGN KEY ('Manid') REFERENCES 'manufactur-
ers' ('Mid') ON DELETE CASCADE ON UPDATE CASCADE;

```

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
<input type="checkbox"/> 1	Wid	int(10)			No	None			Change Drop More
<input type="checkbox"/> 2	Wname	varchar(40)	latin1_swedish_ci		No	None			Change Drop More
<input type="checkbox"/> 3	Wtype	varchar(40)	latin1_swedish_ci		No	None			Change Drop More
<input type="checkbox"/> 4	Wlicense	varchar(20)	latin1_swedish_ci		No	None			Change Drop More
<input type="checkbox"/> 5	Manid	int(10)			No	None			Change Drop More

4.2 Codes Used For Modules:

Insert

```
@app.route('/register',methods=['POST','GET'])
def register():
    if request.method == "POST":
        Uid=request.form.get('uid')
        Uname=request.form.get('uname')
        Uwing=request.form.get('uwing')
        Upass=request.form.get('upass')
        user=Users.query.filter_by(Uid=Uid).first()
        if user:
            alert(text='User ID Already Exists!', title='Message Alert', button='OK')
            # print("Email Already Exist","warning")
            return redirect(url_for('login'))
        encpassword=generate_password_hash(Upass)
        userVendID = 1
        new_user=db.engine.execute(f"INSERT INTO 'users' ('Uid','Uname','Uwing','Upass','Vendid') VALUES ({Uid},{Uname},{Uwing},{encpassword},{userVendID})")
        # newuser=User(username=username,email=email,password=encpassword)
        # db.session.add(newuser)
        # db.session.commit()
        alert(text='Kindly Log-In Yourself Now!', title='Successfully Registered', button='OK')
        # print("Signup Success Please Login","success")
        return redirect(url_for('login')) # return redirect(url_for('login'))
    return render_template('register.html') # return redirect(url_for('register'))
```

Delete

```
@app.route("/delete/<string:Wid>",methods=['POST','GET'])
# @login_required
def delete(Wid):
    db.engine.execute(f"DELETE FROM `weapons` WHERE `weapons`.`Wid`={Wid}")
    alert(text='You\'ve Successfully Deleted The Selected Tuple!', title='Deletion Successful', button='OK')
    return redirect('/weapons')
```

Update

```
@app.route("/edit/<string:wid>", methods=['POST', 'GET'])
# @login_required
def edit(wid):
    posts=Weapons.query.filter_by(wid=wid).first()
    if request.method=="POST":
        Wname=request.form.get('Wname')
        Wtype=request.form.get('Wtype')
        Wlicense=request.form.get('Wlicense')
        # Manid=request.form.get('Manid')
        db.engine.execute(f"UPDATE `weapons` SET `Wname` = '{Wname}', `Wtype` = '{Wtype}', `Wlicense` = '{Wlicense}' WHERE `weapons`.`wid` = {wid}")
        alert(text='You\'ve Successfully Updated The Selected Tuple!', title='Updation Successful', button='OK')
        return redirect('/weapons')
    return render_template('edit.html', posts=posts)
```

Search

```
@app.route('/search', methods=['POST', 'GET'])
# @login_required
def search():
    if request.method=="POST":
        query=request.form.get('search')
        Wname=Weapons.query.filter_by(Wname=query).first()
        Wtype=Weapons.query.filter_by(Wtype=query).first()
        # Wlicense=Weapons.query.filter_by(Wlicense=query).first()
        if Wname:
            query=db.engine.execute(f"SELECT * FROM `weapons` WHERE `weapons`.`Wname`='{query}'")
            alert(text='We Got You Covered!', title='Found!', button='OK')
        elif Wtype:
            query=db.engine.execute(f"SELECT * FROM `weapons` WHERE `weapons`.`Wtype`='{query}'")
            alert(text='We Got You Covered!', title='Found!', button='OK')
        else:
            alert(text='Sorry, Stocks Unavailable Currently!', title='Missing!', button='OK')
            return redirect('/weapons')
    return render_template('weapons.html', query=query) # return redirect(url_for('/weapons', posts=posts)) # render_template('weapons.html')
```

Display

```
@app.route('/details')
#@login_required
def details():
    # posts=Trigr.query.all()
    posts=db.engine.execute("SELECT * FROM `trig`")
    return render_template('triggers.html', posts=posts)
```

Trigger

```
CREATE TRIGGER 'weaponDeletion' BEFORE DELETE ON 'weapons'
FOR EACH ROW INSERT INTO trig
VALUES(null,OLD.Wid,OLD.Wname,OLD.Wtype,OLD.Wlicense,'Weapon
Deleted!',NOW())

CREATE TRIGGER 'weaponInsertion' AFTER INSERT ON 'weapons'
FOR EACH ROW INSERT INTO trig
VALUES(null,NEW.Wid,NEW.Wname,NEW.Wtype,NEW.Wlicense,'Weapon
Inserted!',NOW())

CREATE TRIGGER 'weaponUpdation' AFTER UPDATE ON 'weapons'
FOR EACH ROW INSERT INTO trig
VALUES(null,NEW.Wid,NEW.Wname,NEW.Wtype,NEW.Wlicense,'Weapon Info
Updated!',NOW())
```

Chapter 5

Results and Discussion

Login Page:

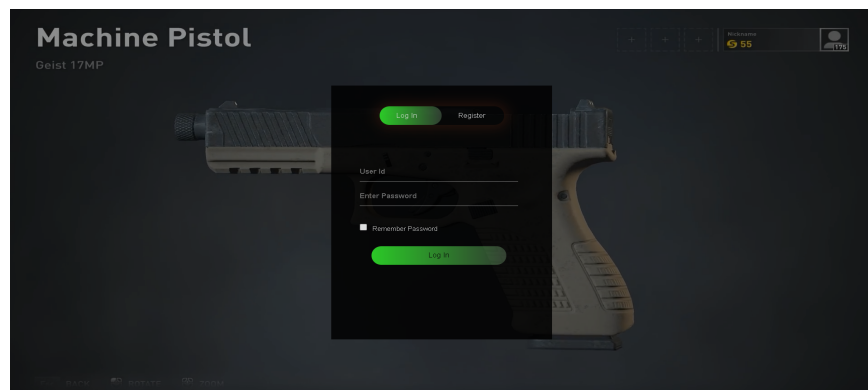


Figure 5.1: Login Page

This is our login page. Here the User can enter his Id and login.

Registration Page:

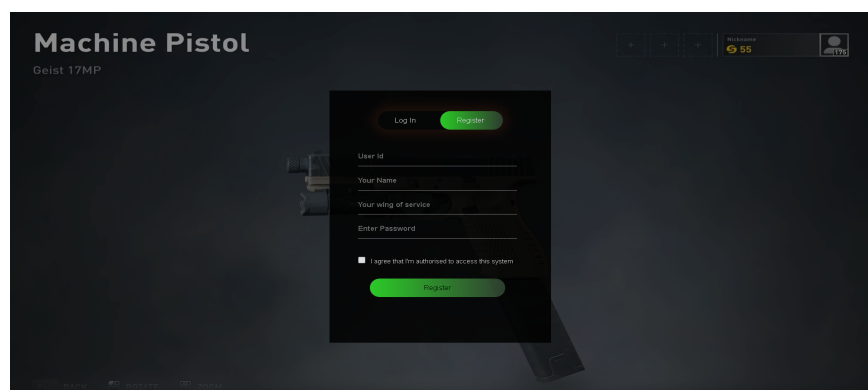


Figure 5.2: Registration Page

This page allows a new User to have access to the database.

Deleting Weapons:

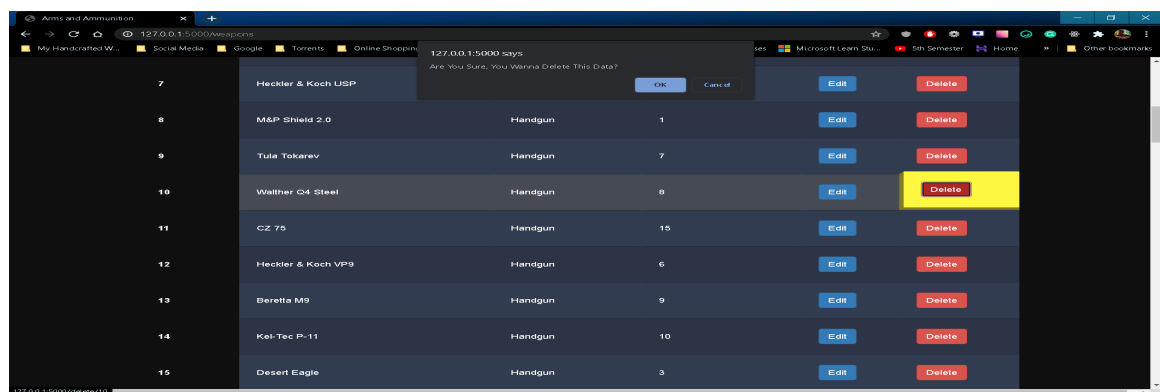


Figure 5.3: Delete Weapons

Here User can delete the weapons of certain manufacturers by selecting that particular row.

Update Weapon Details:

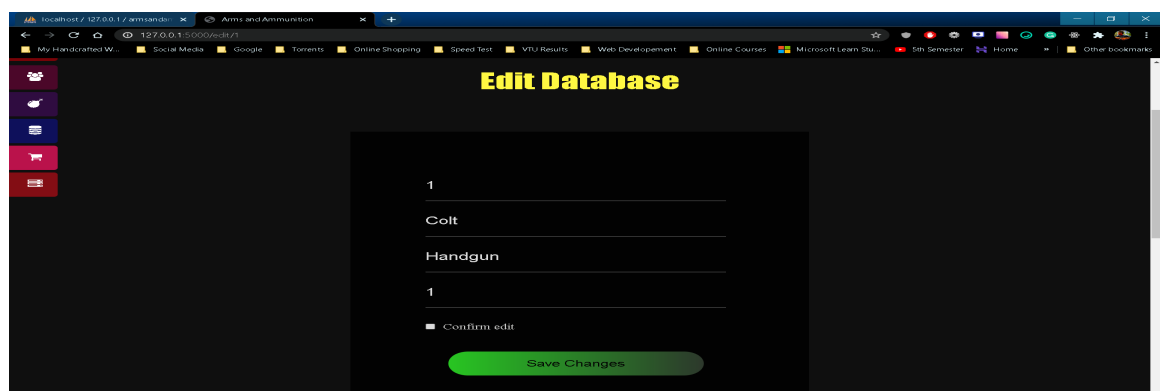


Figure 5.4: Update Weapon Details

Here User can update the details of certain weapons.

Search Weapon Details:

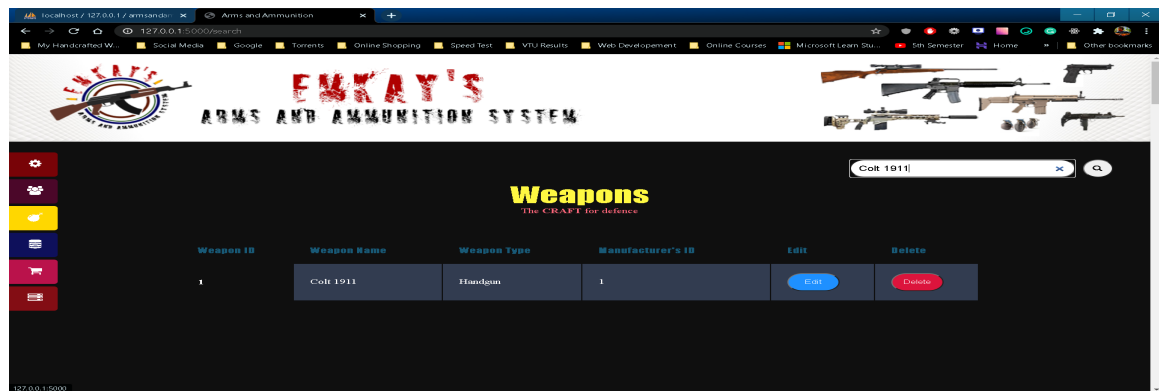


Figure 5.5: Search Weapon Details

Here User can search for the details of certain weapons using type or name of the weapon.

Trigger:

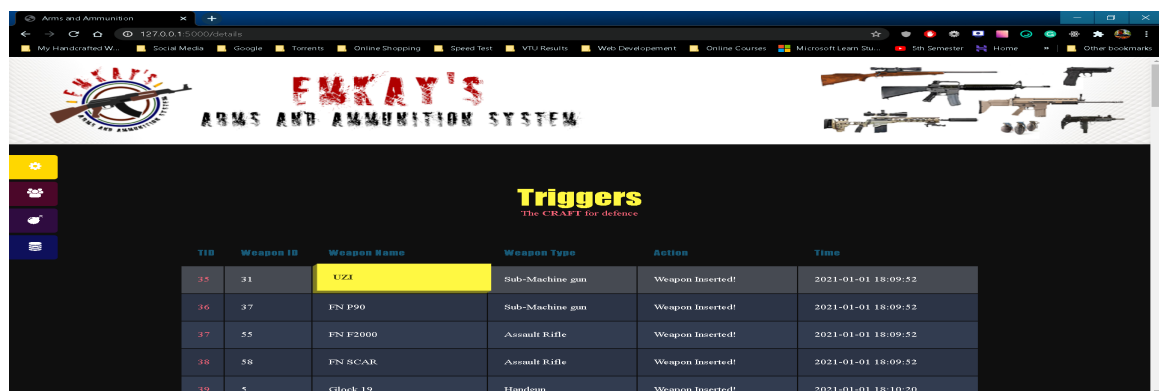


Figure 5.6: Trigger for Modification of Weapons

Trigger will be violated if anyone modifies certain attributes of weapons.

Mail:

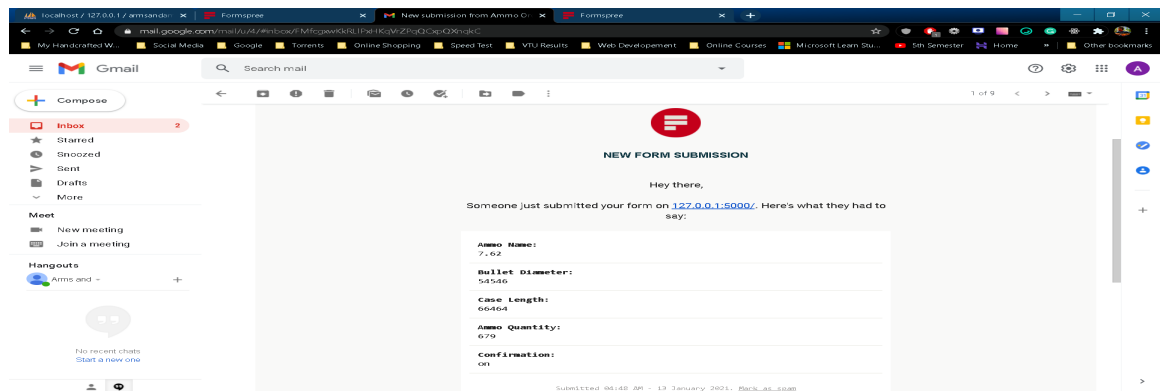


Figure 5.7: Mail Generated

Generating the mail.

Chapter 6

Conclusion and Future work

The small and medium-sized Arms and Ammunition system is using Python-Flask SQLAlchemy Toolkit to develop and realize procurement and inventory management has always been an essential part of the weapons. Arms and Ammunition Management System realizes the function of procurement management, inventory management, sales management, user management and membership management that Army needed. With the continuous improvement of science and technology, the computer's powerful function has been known and used. Compared with the old manual work, the system not only reduces the workload, but also greatly reduced the occurrence of human error. This project can be further improved by differentiating user and admin interfaces so that a recruit might not alter the database and only an experienced officer can administrate the system. Keeping track of when the user or the manufacturer was registered or deleted, giving a warning when the weapon and ammos quantity goes low. Keeping a track of the sales that have happened on a particular day.

References

- [1] Database systems Models, Languages, Design and Application Programming, Ramez Elmasri and Shamkant B. Navathe, 6th Edition, Pearson.
- [2] Database management systems, Ramakrishnan, and Gehrke, 3rd Edition, 2014, McGraw Hill.
- [3] Silberschatz Korth and Sudharshan: Database System Concepts, 6th Edition, McGraw Hill, 2013.
- [4] Coronel, Morris, and Rob, Database Principles Fundamentals of Design, Implementation and Management, Cengage Learning 2012.