**Mavericks**

# Robobees to aid Fruit and Plant Pollination Services

Prepared for:
INFO 6210 – Data Management and Database Design
Spring 2017
Project Report

Guided by:
Professor Chaiyaporn Mutsalklisana


**Team: Mavericks**
Deepa Elangovan (elangovan.d@husky.neu.edu)
Hanan Alsalamah(alsalamah.h@husky.neu.edu)
Mayuresh Mohan Dabholkar(dabholkar.m@husky.neu.edu)
Mrinal Payannavar(payannavar.m@husky.neu.edu)
Yashodhan Vikas Prabhune(prabhune.y@husky.neu.edu)


Northeastern University
College *of* Engineering

# Table of Contents

# Table of Figures

# 1. INTRODUCTION

Roughly three-quarters of the world's flowering plants and about 35 percent of the world's food crops depend on animals to pollinate them, according to the U.S. Department of Agriculture. Some of nature's most prolific pollinators are bees. However, bee populations are declining around the world. A 2016 study looking at 18 years of data on bee populations in England found that insecticides were linked to a long-term decline in the pollinators.

There are many on-going researches studies aiming to address this problem. The most prominent one amongst them is the invention of Robobees led by Professor Robert Wood at Harvard University. Inspired by the biology of a bee, these manmade autonomously flying microbots have the ability to lift off the ground and hover midair. The body development consists of a compact and seamlessly integrated power source to fly on their own, "smart" sensors and control electronics that mimic the eyes and antennae of a bee, and can sense and respond dynamically to the environment. These robots utilize machine learning capabilities and artificial muscles to power and control their flight.

This project aims at facilitating this development to satiate the global crop demands. This involves the design and development of an application which provides an end to end solution for the pollination process carried out by the Robobees. This project employs the database design approach involving the implementation and management of a database that logs the activities of the Robobees throughout its flight and the entire process of pollination carried out by them. The project also involves the design and maintenance of a web based front end interface through which users can place requests and track the pollination process. Furthermore, a relational database management system that can keep track of the movement and status of the bees, will aid in increasing the efficiency if the Robobees.

**Team Information**

Our team is a distinct mix of technologists and strategists. We integrate the team's managerial skills with its technical expertise in the fields of Information Systems, Software Development, Telecommunication and Engineering Management and bring to life the set objectives for the project.

# 2. FEATURES

The application provides a website on which users can register, request for the Robobees, check the progress of the bees and the degree of pollination. It also provides the success rate of the past projects undertaken in different regions. The back end of the application is a database that maintains a log of the users and their information, the Robobees and its status, the drones that control the units of Robobees assigned to them and the crops and plants that have been pollinated.

Some of the prominent features of this application are:

- User-friendly interface
- Centralized repository of all the Robobees
- Continuous monitoring of the status of the Robobees and the pollination process
- Drones to control units of Robobees
- GRP coordinates acquisition
- Statistical analysis to improve performance and success rate
- Data Visualization using BI Tools

# 3. E-R DIAGRAM

The logical structure of the database created for this application is depicted through the E-R Diagram. Fig 5.1 Illustrates the relationships of the entity sets stored in the database. Entities may be characterized not only by relationships, but also by additional properties (*attributes*), which include identifiers called "primary keys".

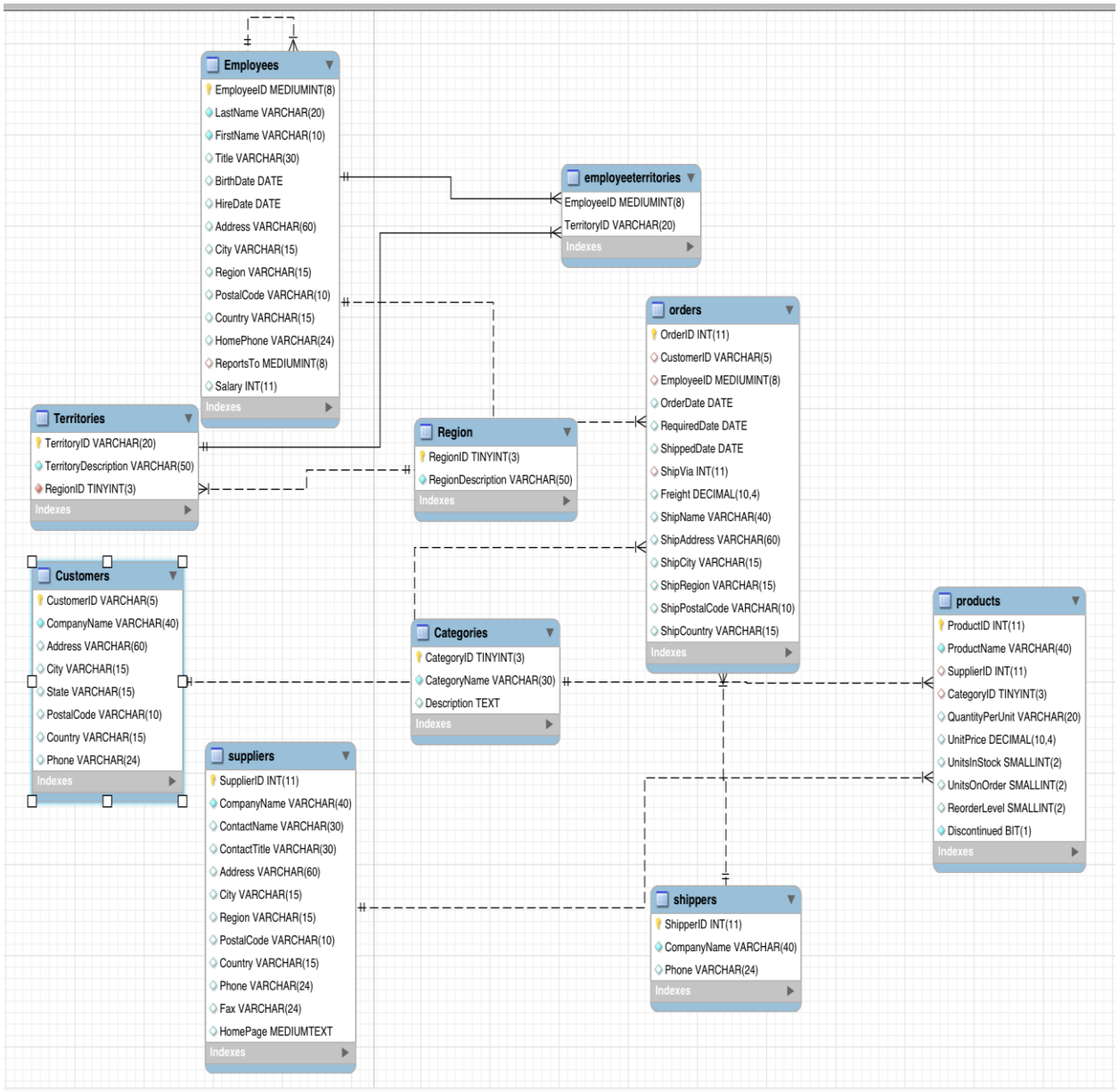



**Fig 3.1: E-R Diagram**

# 4. WORK FLOW

The essential process flow of this application is illustrated through Fig 6.1.

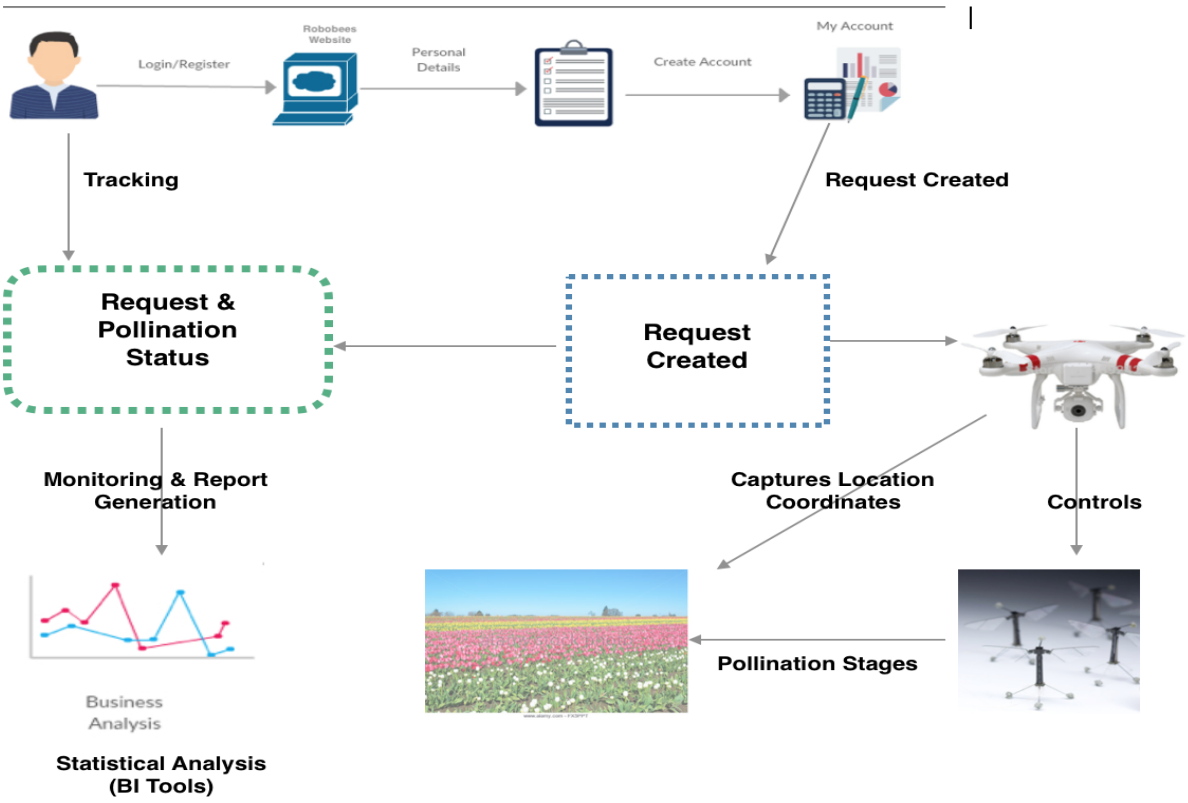



**Fig 4.1: Application Work Flow**

- The application allows users to create accounts through its website. The user enters personal details and a username and password initially.
- Once the user is registered, he/she may create a request for pollination.
- The request is recorded in the application database and a drone is assigned to that request.
- The drone captures the coordinates of the location. Each drone controls a unit of Robobees. Hence, when a drone is assigned to a task, automatically a number of Bees are assigned to that request as well
- The Robobees then resume the pollination process. The in-flight process of the Robobee is continuously monitored by the application
- Hence, users can track their request details and the status of the request through the online portal
- The functionality of the Robobees and its success rate can be generated through statistical analysis
- This is attained using BI tools and is a feature for users to access on the web portal.

# 5. IMPLEMENTATION

## 5.1. User Interface

The User Interface has been designed to simplify the process of pollination. It has been designed using Bootstrap, HTML, CSS, jQuery (for field validations). The interface is easy to navigate, and also includes pages providing information/useful statistics about the pollination process. Some of the pages that are part of the pollination website are:

- Homepage
- Create New Request
- About Us
- Reports

i. **Home Page:** This is the landing page for the Robobee website. It includes an introductory video about Robobees and their applications in the field of pollination.

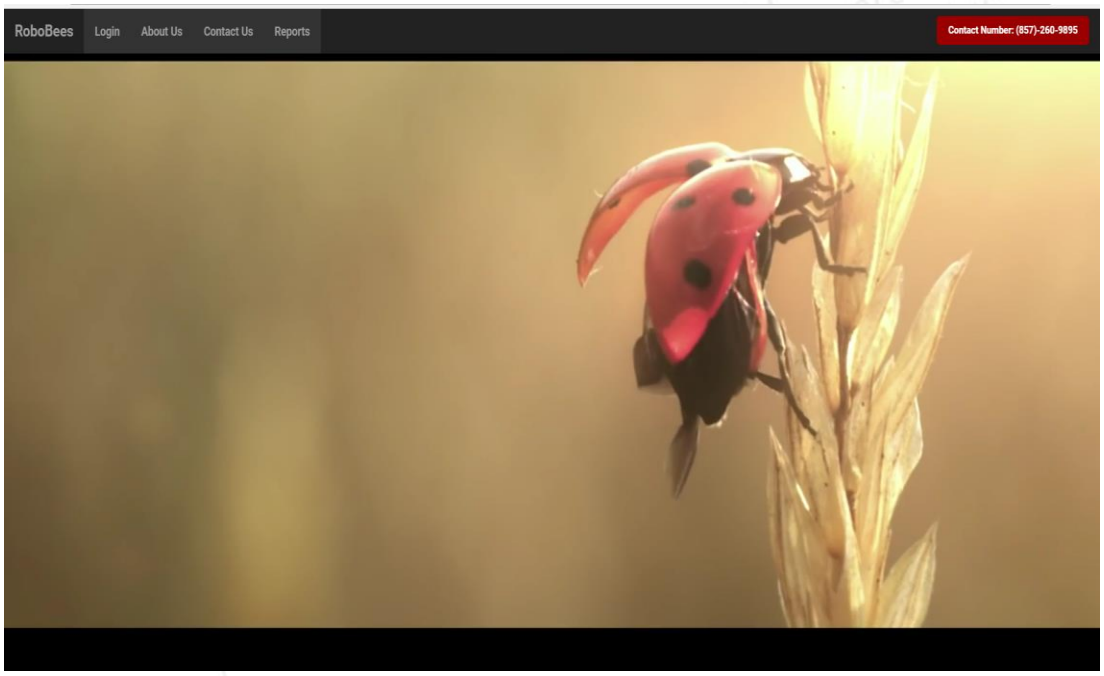



**Fig 5.1.1: Homepage**

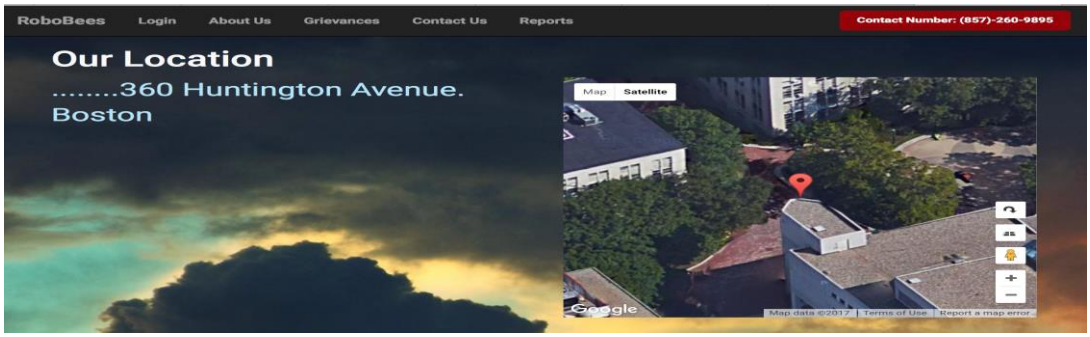



**Fig 5.1.2: Contact**

ii. **Login:** The Homepage includes a login modal for single sign on and new user registration. This is designed with CSS modal and it will navigate the user to his home page or to the registration page in case of new user. In case of user input mismatch in username and password system will give a message as “Incorrect login information”.

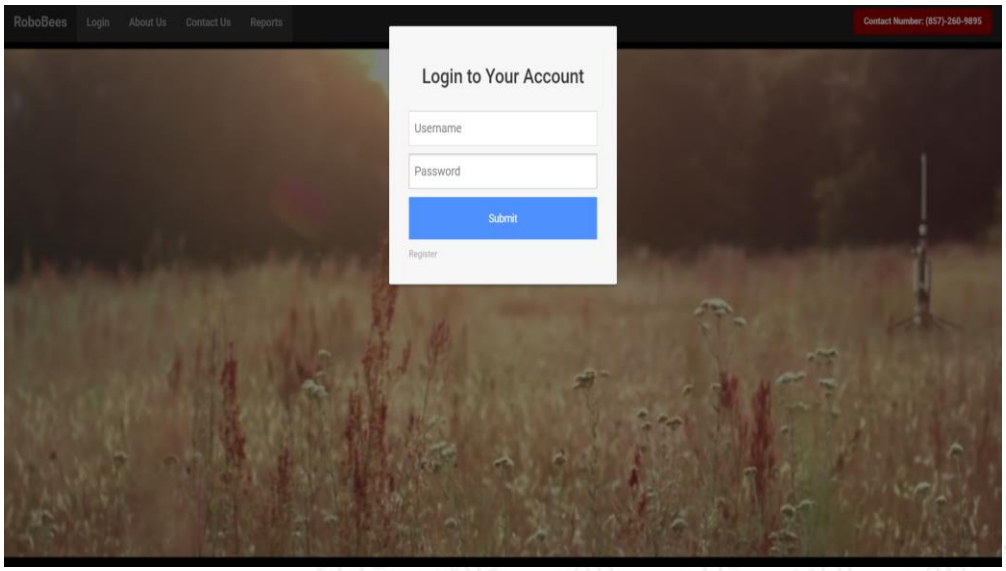



**Fig 5.1.3: Login Model**

iii. **Register Page**: The login modal also has a link for the user to register himself on the website. Once registered, he can login to create new requests.

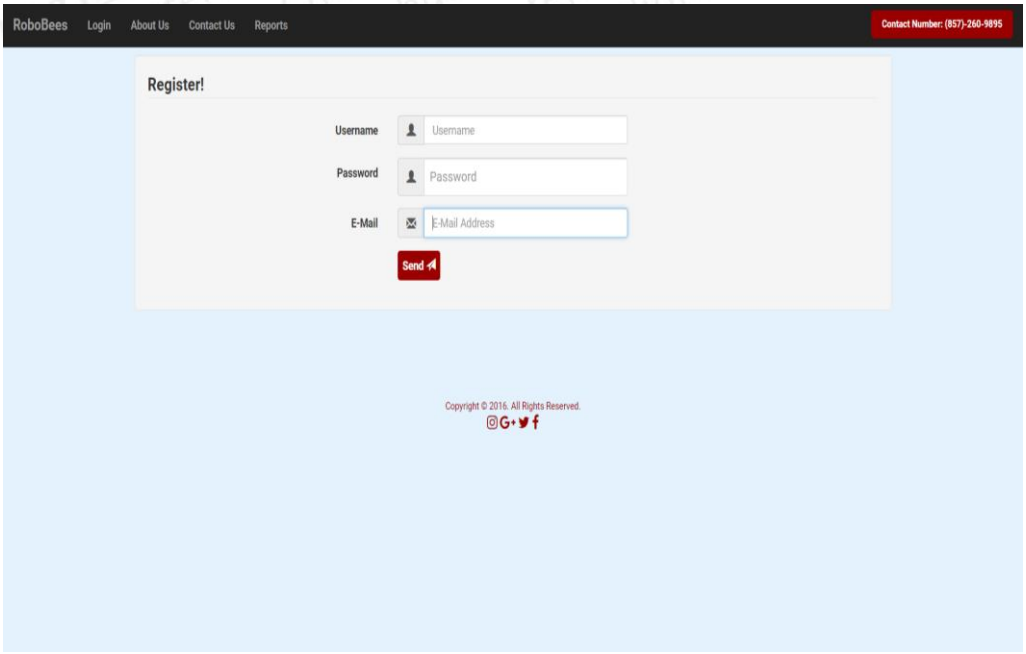



**Fig 5.1.4: Registration page**

iv. **Create New Request:** After the user has successfully logged in to the website, he/she will be redirected to the create new request page. Here, the user will have the option to create a new request for pollination and specify information related to the field (area of field, crop/flower type) on which pollination is to be performed.

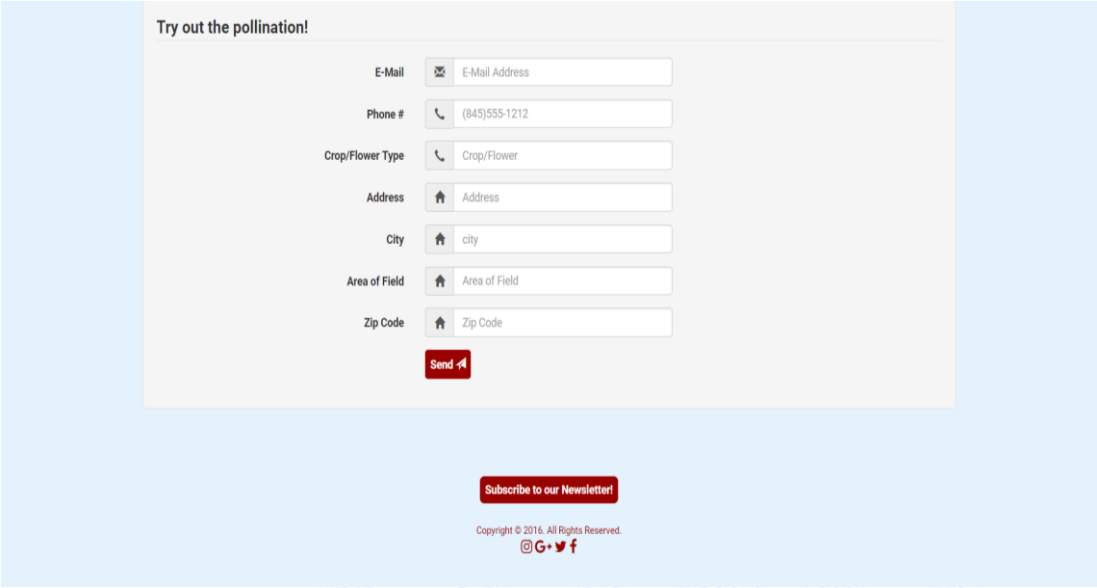



**Fig 5.1.5: Create new request**

v. **About Us:** Apart from these pages, there is also an About Us page which provides information regarding the pollination process. It shows statistics about the number of fields pollinated, number of bees and drones that are currently involved in the process of pollination.

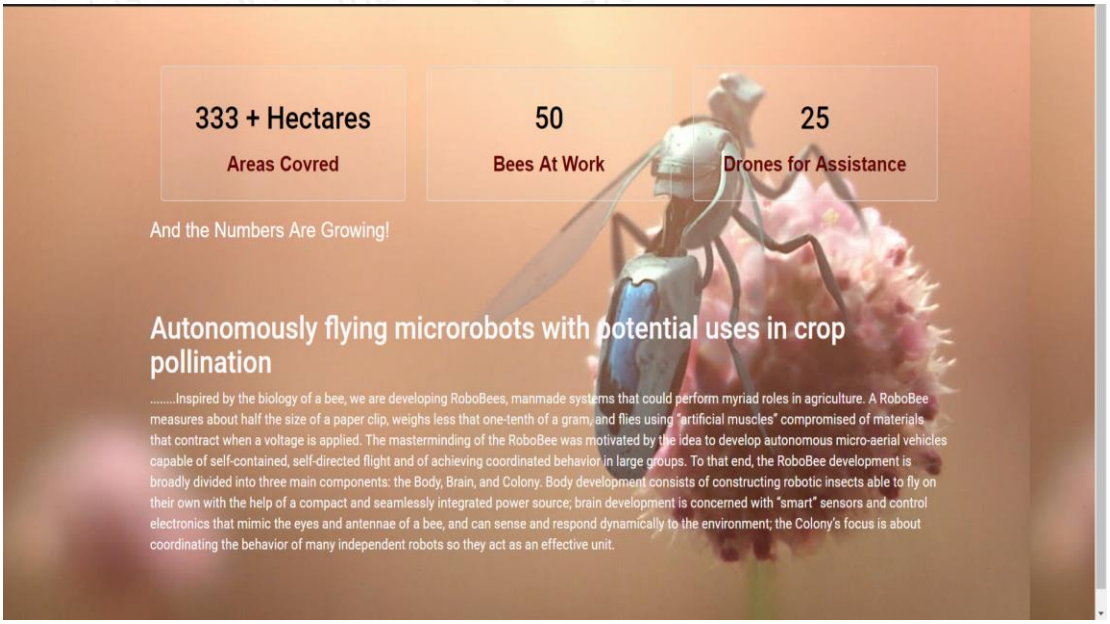



**Fig 5.1.6: About Us**

**vi. Reports:** There is also a Reports page where the admin can login to view statistics related to the pollination process. The Reports have been generated using the PowerBI Business Intelligence tool.

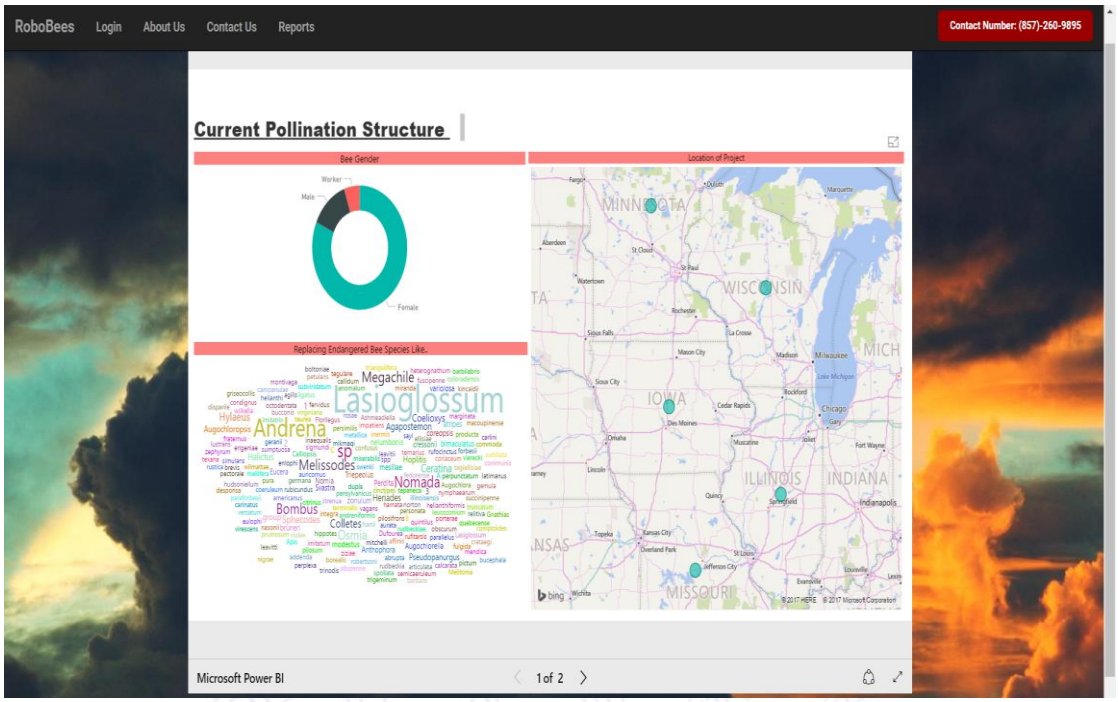



**Fig 5.1.7: Reports using PowerBI**

## 5.2. Back End Connectivity

Our website is connected to a MySQL Database. Operations performed using the user interface will be reflected in the database. This connectivity has been achieved using PHP, which is bundled as part of the WAMP Server.

```php
<?php


   session_start();
     $username = "root";
     $password = "root";
     $host = "localhost";
     $dbname ="projectfinal";

     // Create connection
$conn = mysqli_connect($host, $username, $password, $dbname);
// Check connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}
```

**Fig 5.2.1: PHP code for connecting to the database**

```php
  if($_SERVER["REQUEST_METHOD"] == "POST") {
     // username and password sent from form

     $myusername = mysqli_real_escape_string($conn,$_POST['user']);
     $mypassword = mysqli_real_escape_string($conn,$_POST['pass']);

     $sql = "SELECT admin FROM User WHERE username = '$myusername' and password = '$mypassword'";
     $result = mysqli_query($conn,$sql);
     //$row = mysqli_fetch_array($result,MYSQLI_ASSOC);
    $row = mysqli_fetch_array($result);
   $admin = $row['admin'];

     $count = mysqli_num_rows($result);

     // If result matched $myusername and $mypassword, table row must be 1 row
     if($count ==1){

     if($admin == 'user') {



        $_SESSION["login_user"] = "$myusername";
        header("location: user-login.html");


     }else if($admin == 'admin') {

          $_SESSION["login_user"] = "$myusername";
         header("location: admin-login.php");


     }
     }
     else{
        header("location: invalidCredentials.html");
     }
  }
?>
```

**Fig 5.2.2: PHP code for implementing login functionality**

# DATABASE IMPLEMENTATION

Once the user creates a new request for pollination, the request gets added to the Request table. After this, to demonstrate the working of the Robobees and the drones, we have written a number of stored procedures, triggers and events. Each of these aspects are described in the section below.

## 5.3.1. Stored Procedure

A stored procedure is a set of Structured Query Language statements with an assigned name that's stored in the database in compiled form so that it can be shared by several programs. Stored procedures work much like functions, and can be used to perform a set of operations on tables. We have used the following stored procedures in our database.

- **Stored procedure to update the status of the Robobee to 'Busy' from 'Available'**

```
39
40    DELIMITER $$
41    create procedure insert_drone2()
42    BEGIN
43    declare x int;
44    set x = 1 ;
45       while (x < 50) do
46       insert into robobee
47         SET Stat='busy',
48         DroneId = (
49         SELECT DroneId from Drone where Stat='working'),
50         RequestId = (
51         SELECT RequestId from Request where status='Pending approval');
52         set x=x+1;
53       end while;
54    END$$
55    DELIMITER $$
56
```

**Fig 5.3.1.1: Procedure to Update Robobee Status**

The above procedure creates a set of Robobees based on user request. It also sets the status for these Robobees to 'busy' and associates them with the Drone and Request to which they belong.

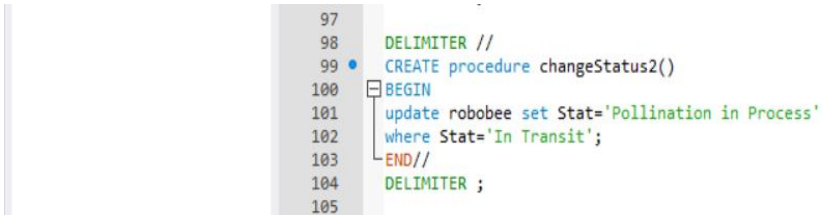
- **Stored Procedure to update the status of the Robobee to 'In Transit' from 'Busy'**

```
89
90    DELIMITER //
91    CREATE procedure changeStatus1()
92    BEGIN
93    UPDATE robobee SET Stat='In Transit'
94    WHERE Stat='busy';
95    END //
96    DELIMITER ;
97
```

**Fig 5.3.1.2: Procedure to Update Robobee Status -2**

The above store procedure changes the status of the Robobee to 'In Transit' from 'Busy'. This is an important stored procedure because it allows us to demonstrate the working of the Robobee, and its transition through the various stages.

- **Stored Procedure to update the status of the Robobee to 'Pollination in Process' from 'In Transit'**

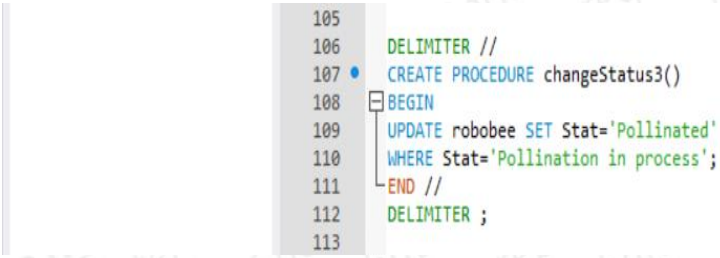


```
97
98   DELIMITER //
99   CREATE procedure changeStatus2()
100  BEGIN
101  update robobee set Stat='Pollination in Process'
102  where Stat='In Transit';
103  END//
104  DELIMITER ;
105
```

**Fig 5.3.1.1: Procedure to Update Robobee Status -3**

This stored procedure changes the status from 'In Transit' to 'Pollination in Process'. It helps in representing the physical activity of the bees pollinating particular flowers in a field. These statuses can be utilized to monitor the progress of the Robobees.

- **Stored Procedure to set the status of the Robobee to 'Pollinated'**

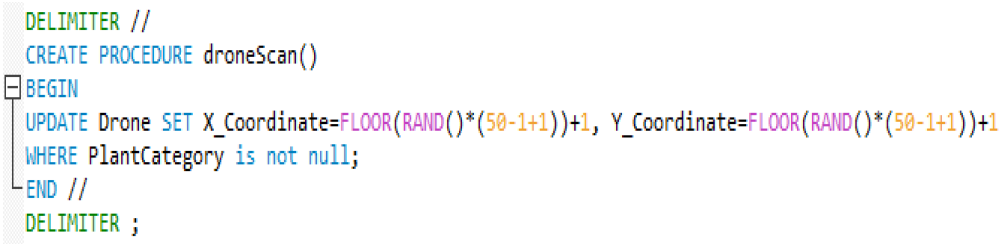


```
105
106  DELIMITER //
107  CREATE PROCEDURE changeStatus3()
108  BEGIN
109  UPDATE robobee SET Stat='Pollinated'
110  WHERE Stat='Pollination in process';
111  END //
112  DELIMITER ;
113
```

**Fig 5.3.1.1: Procedure to Update Robobee Status-4**

The above stored procedure sets the status of the Robobees to 'Pollinated', thus representing the completion of the pollination process for a particular request. Any subsequent request which comes in, will be allocated to a different drone, and to a different set of Robobees associated with that drone.

- **Stored procedure to generate location co-ordinates from Drones**



```
DELIMITER //
CREATE PROCEDURE droneScan()
BEGIN
UPDATE Drone SET X_Coordinate=FLOOR(RAND()*(50-1+1))+1, Y_Coordinate=FLOOR(RAND()*(50-1+1))+1
WHERE PlantCategory is not null;
END //
DELIMITER ;
```

**Fig 5.3.1.1: Procedure to Generate Coordinates from Drone**

The above stored procedure generates random coordinates at specific intervals for all the drones currently employed by Robobee application. These coordinates represent the coordinates for the flowers, which will then be assigned to individual Robobees. The Robobees will then navigate to these coordinates and perform the process of Pollination.

### 5.3.2. Triggers

A trigger is a special kind of stored procedure that automatically executes when an event occurs in the database server. DML triggers execute when a user tries to modify data through a data manipulation language (DML) event. DML events are INSERT, UPDATE, or DELETE statements on a table or view.

**Advantages of Triggers**

- Triggers can be written for the following purposes:
- Generating some derived column values automatically
- Enforcing referential integrity
- Event logging and storing information on table access
- Preventing invalid transactions
- We have written the following triggers for our database implementation

- **Trigger to create a new drone whenever a new request is generated**

```
12
13  DELIMITER $$
14  create trigger after_request_insert
15  AFTER INSERT on Request
16  for each row
17  begin
18  INSERT INTO Drone
19     SET Stat = 'working',
20         RequestId = (
21         SELECT RequestId
22           FROM Request
23          WHERE status = 'pending approval');
24  end $$
25  DELIMITER ;
26
```

**Fig 5.3.2.1: New Drone Trigger**

Whenever a new pollination request comes in, a specific drone gets assigned to attend and resolve the request. This function has been demonstrated by using triggers. The drone status will be set to 'working'.

- **Trigger to call a stored procedure**

```
30
31  DELIMITER $$
32  create trigger after_drone_insert1
33  after insert on Drone
34  for each row
35  begin
36  call insert_drone2();
37  end$$
38  DELIMITER $$
39
```

**Fig 5.3.2.2: Trigger to call procedure**

The above trigger runs after a drone is inserted in the drone table. This trigger will then call the stored procedure, that will create Robobees. Thus, it helps the simulating the process of drone and Robobee creation.

### 5.3.3. Events

MySQL Events are tasks that run according to a schedule. Therefore, we sometimes refer to them as scheduled events. When you create an event, you are creating a named database object containing one or more SQL statements to be executed at one or more regular intervals, beginning and ending at a specific date and time. Conceptually, this is similar to the idea of the Unix crontab (also known as a "cron job") or the Windows Task Scheduler. We have used the following events in our database application.

- **Event to call a stored procedure 'changeStatus1()'**

```
57
58  set global event_scheduler = on;
59  DELIMITER $$
60  CREATE EVENT a_update
61  ON SCHEDULE AT CURRENT_TIMESTAMP + INTERVAL 10 SECOND
62  DO
63  BEGIN
64    call changeStatus1();
65  END $$
66  DELIMITER ;
67
```

**Fig 5.3.3.1: Change Status Event Handler – 1ST Interval**

The above event will consider the current timestamp, add an interval of 10 seconds and then call the stored procedure changeStatus1() – which will update the status of the Robobee to 'In Transit'.

- **Event to call a stored procedure 'changeStatus2()'**

```
67
68  set global event_scheduler = on;
69  DELIMITER $$
70  CREATE EVENT b_update
71  ON SCHEDULE AT CURRENT_TIMESTAMP + INTERVAL 20 SECOND
72  DO
73  BEGIN
74    call changeStatus2();
75  END $$
76  DELIMITER ;
```

**Fig 5.3.3.2: Change Status Event Handler – 2nd Interval**

The above event will call the procedure changeStatus2() – which updates the status of the Robobee to 'Pollination in Process'.

- **Event to call a stored procedure 'changeStatus3()'**

```
77
78  set global event_scheduler = on;
79  DELIMITER $$
80  CREATE EVENT c_update
81  ON SCHEDULE AT CURRENT_TIMESTAMP + INTERVAL 30 SECOND
82  DO
83  BEGIN
84    call changeStatus3();
85  END $$
86  DELIMITER ;
```

**Fig 5.3.3.3: Change Status Event Handler– 3rd Interval**

The above event will call the procedure changeStatus3() – which updates the status of the Robobee to 'Pollinated'. These events enable us to replicate the functioning of an actual Robobee, and help us to demonstrate the process of pollination.

- Event to call the droneScan() procedure

```
# The drone will scan the field every 5 second
set global event_scheduler = on;
DELIMITER $$
CREATE EVENT generateCoordinates
ON SCHEDULE AT CURRENT_TIMESTAMP + INTERVAL 5 SECOND
DO
BEGIN
  call droneScan();
END $$
DELIMITER ;
```

**Fig 5.3.3.4: Event Handler– Generate Coordinates**

This procedure has been implemented to execute after 5 seconds from the current timestamp. It calls the droneScan() stored procedure, which generates location coordinates for the flowers.

### 5.3.4. Privileges

The privileges granted to a MySQL account determine which operations the account can perform. MySQL privileges differ in the contexts in which they apply and at different levels of operation:

- Administrative privileges enable users to manage operation of the MySQL server. These privileges are global because they are not specific to a particular database.
- Database privileges apply to a database and to all objects within it. These privileges can be granted for specific databases, or globally so that they apply to all databases.
- Privileges for database objects such as tables, indexes, views, and stored routines can be granted for specific objects within a database, for all objects of a given type within a database (for example, all tables in a database), or globally for all objects of a given type in all databases).

We have implemented the following privileges in our database application.

```
create user newuser identified by '123a';
grant select on robobee to newuser;

grant select on request to newuser;
grant update(status) on request to newuser;

grant select on flower to newuser;
grant update (flowername) on flower to newuser;

grant insert on flower to newuser;
Revoke insert on flower from newuser;
```

**Fig 5.3.4.1: Privileges**

These privileges ensure that access to the database is restricted only to the users who are authorized to make changes to the database. An admin is allowed to make changes (Update/delete requests), but a user ('newuser' in this case) is only allowed to view his/her requests.

### 5.3.5. Other SQL Operations

In addition to the operations mentioned above, our application also includes the use of Subqueries, Joins, Views and Transactions. These operations are defined below.
**Subqueries:** A subquery is a SQL query nested inside a larger query.

- A subquery may occur in:
- A SELECT clause
- A FROM clause
- A WHERE clause
- In MySQL, subquery can be nested inside a SELECT, INSERT, UPDATE, DELETE, SET, or DO statement or inside another subquery.
- A subquery is usually added within the WHERE Clause of another SQL SELECT statement.

**Select * from robobee where requestId=(select RequestId from request where username="yash");**

The above subquery allows the user to track all the RoboBees currently working on his/her request and also see their status.

**Views:** A database view is a virtual table or logical table which is defined as a SQL SELECT query with joins. Because a database view is similar to a database table, which consists of rows and columns, so you can query data against it. Most database management systems, including MySQL, allow you to update data in the underlying tables through the database view with some prerequisites.

**CREATE VIEW request_view AS**
**SELECT request.RequestId, request.status, robobee.DroneId, robobee.Stat**
**FROM request JOIN robobee ON request.RequestId= robobee.RequestId;**

**SELECT * FROM request_view;**

The above query creates a view called "request_view". It displays the requests that are currently in progress, alongwith the RoboBees who are working on the request.

**Transactions:** A transaction is a unit of work that is performed against a database. Transactions are units or sequences of work accomplished in a logical order, whether in a manual fashion by a user or automatically by some sort of a database program. We have used the COMMIT and START TRANSACTION commands while inserting and updating values in the database.

### 5.3.6. Database Backup

It is important to back up your databases so that you can recover your data and be up and running again in case problems occur, such as system crashes, hardware failures, or users deleting data by mistake. To ensure that our database is protected, we have used the following backup strategy using mysqldump. In addition, all the data that is sent to PowerBI is automatically backed up.

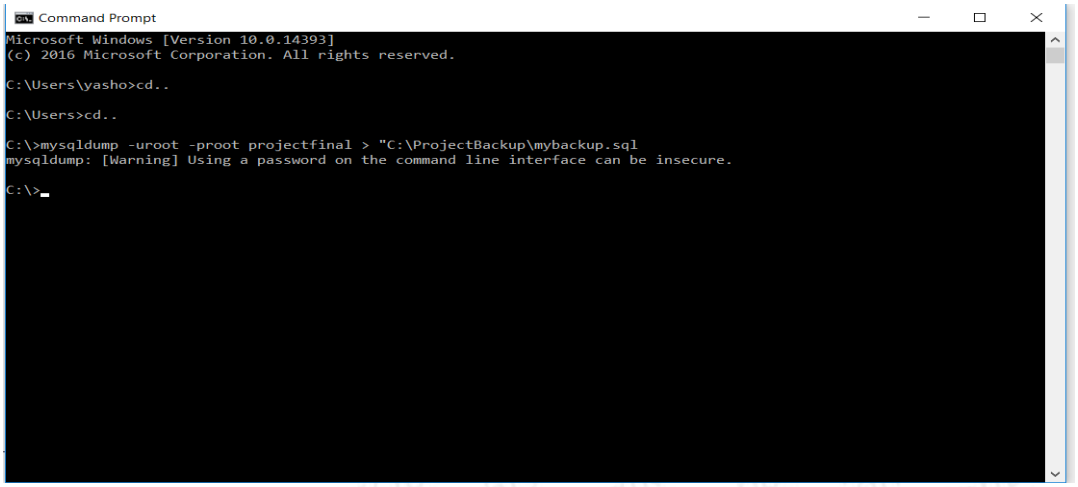



**Fig 5.3.6.1: Backup Process**

Once this command is executed, the backup file is created at the location mentioned in the command –

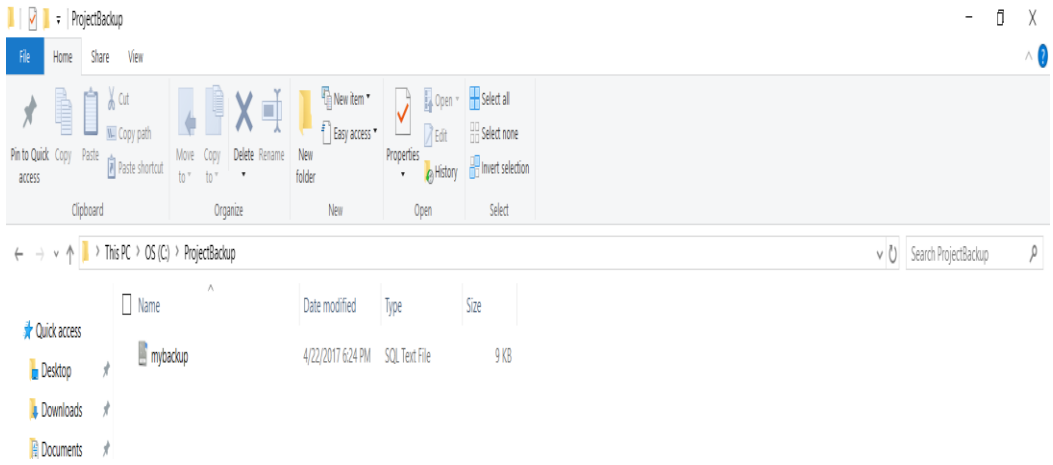



**Fig 5.3.6.2: Backup File**

## 5.4. BI ANALYSIS

Tolls used: Microsoft Power BI

We have implemented a Power BI analysis of bees and the ongoing project status. Data visualization helps to display information and aids in future data analysis. In this application, it will be helpful for users to analyze the effectiveness of the overall process of pollination using Robobees.

1. We have conducted an analysis of the existing the average number of bees that are currently under the threat of extinction, this classification is done with respect of their gender

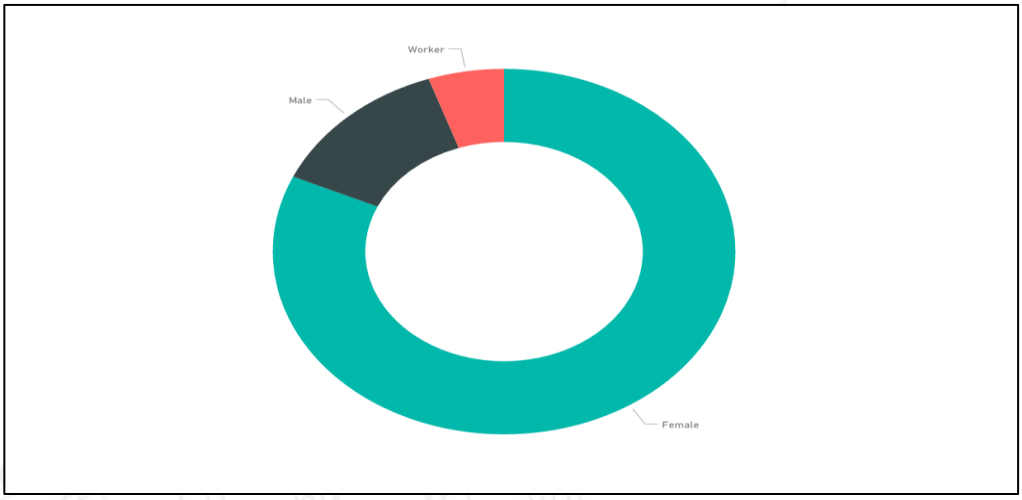



**Fig 5.4.1: Analysis Chart**

2. We have incorporated the status of ongoing projects and their location using the Bing Maps.

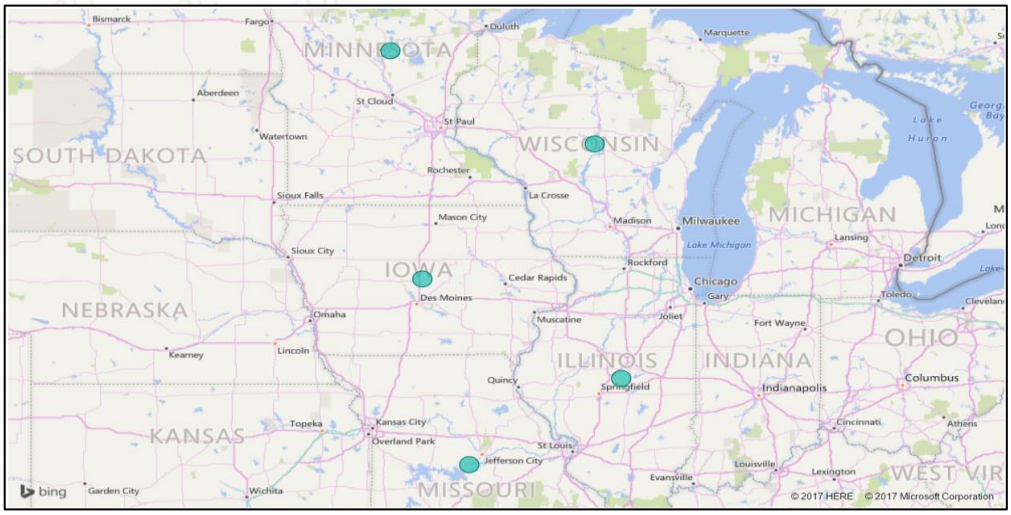



**Fig 5.4.2: Map View**

3. A word cloud of all the species that are currently being replaced by artificial bees.


**Fig 5.4.3: Word Cloud - Species**

4. The above charts are placed together so as to give user the liberty to select a particular location or species or gender and get a deeper analysis of other two matrices.

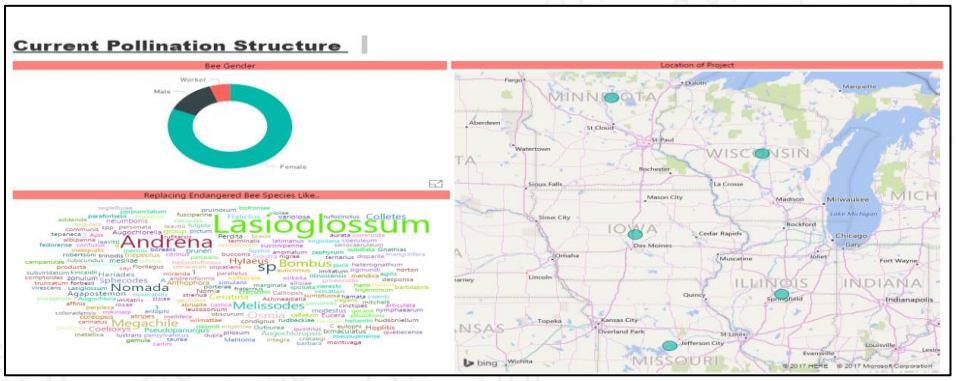



**Fig 5.4.4: Current Pollination Structure**

Other Parts of the analysis is that of successful pollination and the hardware used in various projects. And most importantly, the Success Rate.

Successful pollination of different flower species
This was shown using the word cloud. It served two purpose.

1. It depicted the total volume (scale) at which the projects were successful.
2. It helped in giving names of individual species that were successfully pollinated using artificial bees.


**Fig 5.4.5: Successful Pollination-Word Cloud**

Next Analysis was that of the average completion rate of current projects.

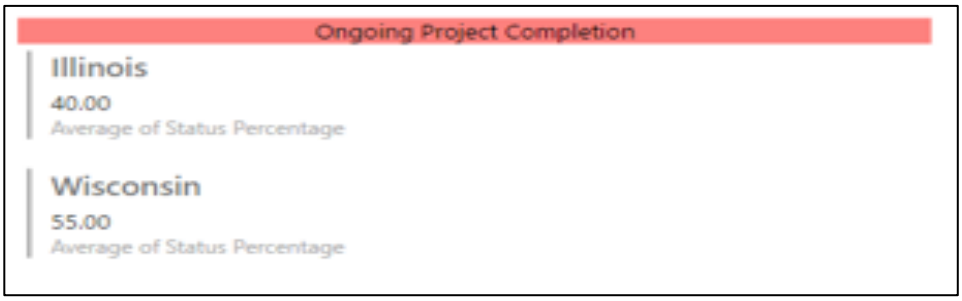



**Fig 5.4.6: Average Completion Rate**

The last analysis was done using the hardware that was used to make the project successful.

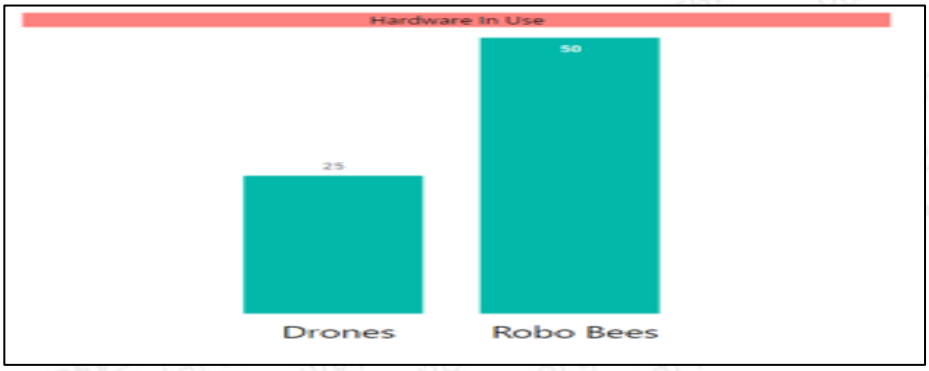



**Fig 5.4.7: Hardware in Use**

Above individual analysis was incorporated together in one dashboard so as to make it more responsive. This gave user the ability to select individual matrices and make it dive deeper into other two matrices.

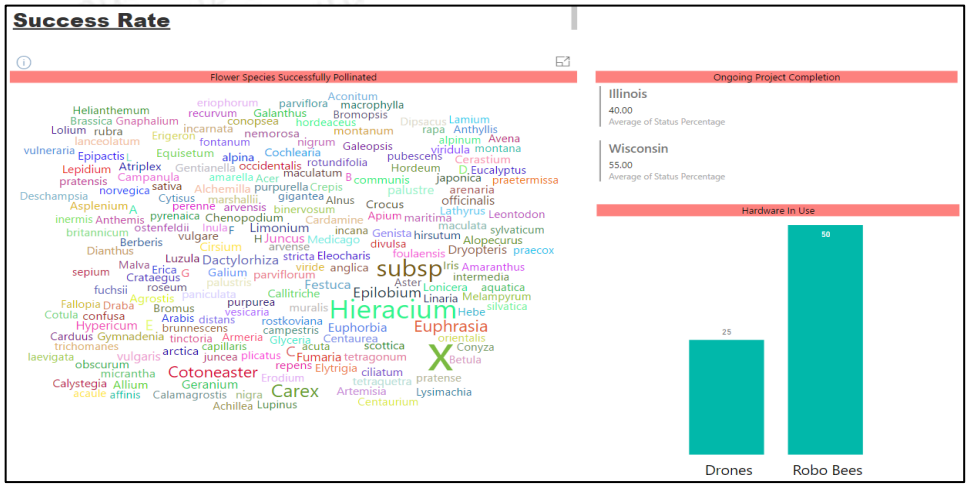



**Fig 5.4.8: Success Rate**

## 5.5. CLOUD DEPLOYMENT

Cloud Computing provides a simple way to access servers, storage, databases and a broad set of application services over the Internet. Cloud Computing providers such as Amazon Web Services own and maintain the network-connected hardware required for these application services, while you provision and use what you need via a web application.

**Why Cloud?**

1. Trade capital expense for variable expense
2. Benefit from massive economies of scale
3. Stop guessing capacity
4. Increase speed and agility
5. Stop spending money on running maintaining data centers
6. Go global in minutes

Robobees application is deployed on amazon AWS cloud instance for high availability and scalability. We aim to use apache tomcat8 as web container and deployment manager to manage our application. A jar source code is placed and extracted in /webapps folder. Also, a database instance is configured on same EC2 instance to maintain DB processing on cloud.

**Cloud Details:**

**Web application Link:**
www.robobees.com.s3-website-us-west-2.amazonaws.com

**Instance Details:**
**Public IP**
54.187.221.118
**Private IPs**
172.31.40.106
**Public DNS**
ec2-54-187-221-118.us-west-2.compute.amazonaws.com

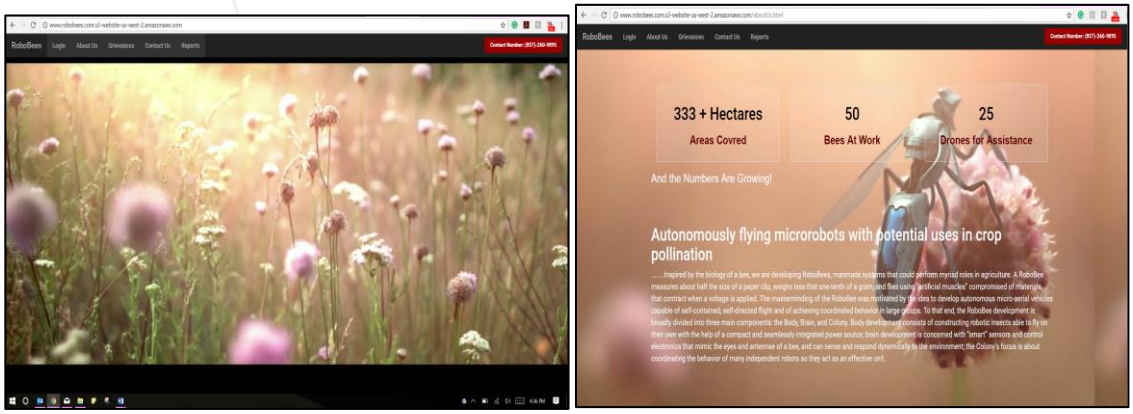



**Fig 5.5.1: Webpage**

# 6. ASSUMPTIONS & CONSTRAINTS

## 6.1. Assumptions

- The project will include a fully functional autonomous swarm Robobees to perform artificial pollination.
- The hive must be resilient enough so that the group can complete its objectives even if many bees fail.
- Robobees are enabled with machine learning capabilities.
- Technology for in air operations of Robobees without tethers is developed and successful.
- Each Robobee has a GPS to record the coordinates of the pollinated flower/crop in order to avoid re-visiting the same flower/crop again.
- Under windy conditions, the flower/crop might change its coordinates. In such a case, the drone will scan the field every 5 seconds by taking an image showing crop/flower coordinates. This will be accomplished through scheduling an event (every 5 seconds) that will call a procedure uses Rand () function for generating random X and Y coordinates.

## 6.2. Constraints and Limitations

- Robobee can learn how to swim, but there's still a big gap in its abilities as it cannot effectively see.
- High cost of robotic systems.
- Limited Battery Life of Robobees: one of the most challenging aspects of the Robobee is its power system design since Robobees have to perform long endurance operations.
- Flight is energy-intensive, and the limitations of current energy storage technologies severely curtail in-air operations.
- Can break easily: the tiny size of Robobees make them fragile to endure various physical disasters.

# 7. Project Management Process

## 7.1 Integrated Master Schedule/Milestones

A detailed schedule has been created to understand the key milestones during the project implementation process. The milestones have been marked to ensure the project was on schedule. The Gantt chart of schedule is depicted below in Figure 6.1.

**Table 7.1: Schedule of Project Milestones**

| Sl. No. | Milestones | Completion Date |
|---|---|---|
| 1. | Setting Project Deliverables & Initial Project Plan | 03/10/2017 |
| 2. | Detailed implementation plan & ERD | 03/17/2017 |
| 3. | Database Design and creation | 03/27/2017 |
| 4. | Testing & Improving database | 03/31/2017 |
| 5. | Web Application Development(Front – End) | 03/24/2017 |
| 6. | Testing Entire Application | 04/03/2017 |
| 7. | Documentation | 04/07/2017 |

The Gantt chart depicts the Schedule of the project based on tasks and deliverables of the projects. It provides the Project timeline based on the start and end dates.

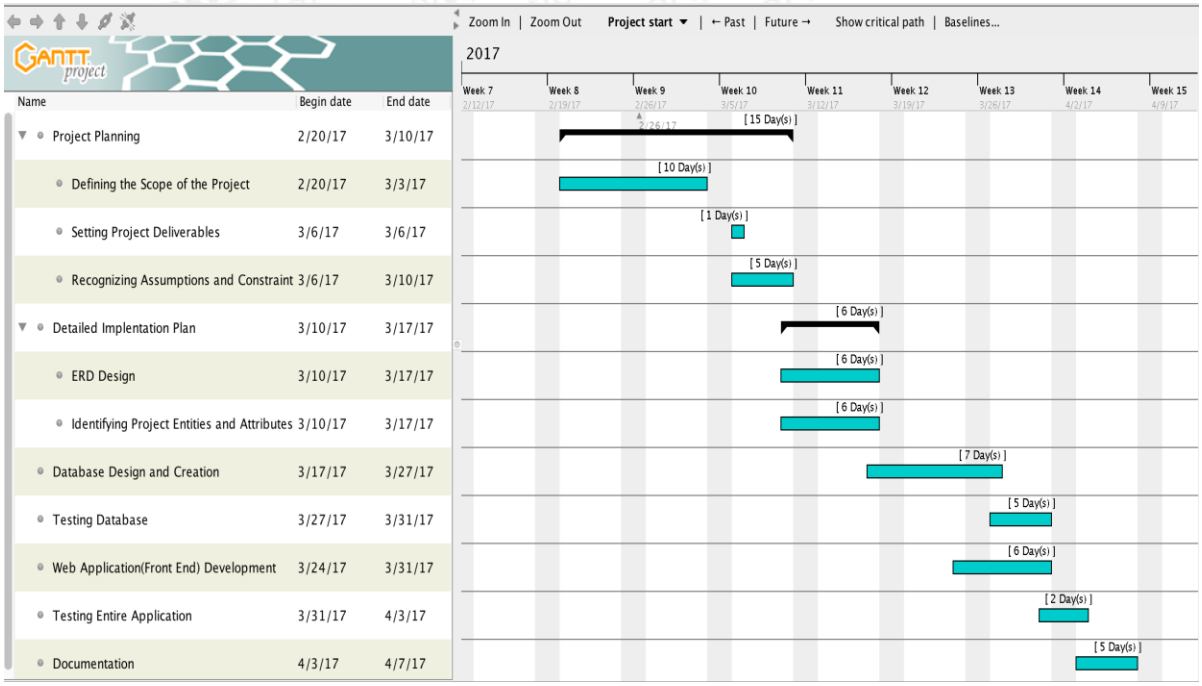



**Fig 7.1.1: Gantt Chart – Schedule**

## 7.2 Project Timeline

In order to ensure the application was designed, developed and tested as per the set schedule, a project timeline was maintained throughout the implementation process. Figure 7.2.1 illustrates the project timeline.

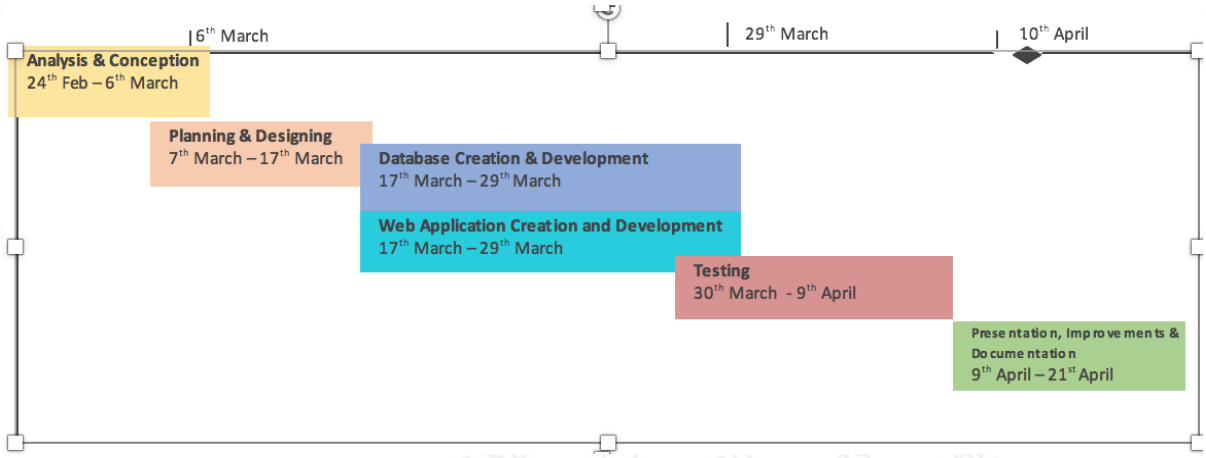



**Fig 7.2.1: Timeline**

## 7.3 Responsibility Chart

The Responsibility Chart below describes the participation of team members in completing tasks or deliverables for the project. The table 4.3.1 illustrates the team members involved in the project and their roles and responsibilities.

**Table 7.3.1: Project Team & Responsibilities**

| Sl No. | Name | Role & Responsibilities |
|---|---|---|
| 1. | Deepa Elangovan | Database Creation & Design ( Triggers, Event Handlers, Stored Procedures) |
| 2. | Hanan Alsalamah | Database Creation ( Stored procedures, Privileges, triggers) |
| 3. | Mayuresh Mohan Dabholkar | Web Application Development, Data analysis using BI |
| 4. | Mrinal Payannavar | Requirements Gathering, Views creation |
| 5. | Yashodhan Vikas Prabhune | Web Application Development, Database design ( Stored Procedures) |

# 8 REFERENCES


1. Rsi. "Overview of the Harvard Robobee Project (Greg Morrisett, Harvard)." *Imperial College London*. N.p., n.d. Web. 15 Mar. 2017.

2. Jennifer Doody, Harvard Staff Writer |, Rachael Dane Harvard Correspondent |, Christina Pazzanese, Harvard Staff Writer |, Peter Reuell, Harvard Staff Writer |, Sue McGreevey, MGH Public Affairs |, Alvin Powell, Harvard Staff Writer |, and Caroline Perry SEAS Communications |. "Robotic insects make first controlled flight." *Harvard Gazette*. N.p., 02 May 2013. Web. 15 Mar. 2017.

3. "Autonomous Flying Microrobots (RoboBees)." Wyss Institute. N.p., 07 Sept. 2016. Web. 15 Mar. 2017.

4. Gent, Edd. "Robo-Bees Could Aid Insects with Pollination Duties." LiveScience. Purch, 09 Feb. 2017. Web. 15 Mar. 2017.

5. "National Science Foundation - Where Discoveries Begin." Flight of the RoboBee | NSF - National Science Foundation. N.p., n.d. Web. 15 Mar. 2017.

6. Donahue, Michelle Z. "RoboBees Can Fly and Swim. What's Next? Laser Vision." Smithsonian.com. Smithsonian Institution, 17 Nov. 2015. Web. 15 Mar. 2017.

7. Wigerman, Elena. "RoboBees." RoboBees. N.p., 01 Jan. 1970. Web. 15 Mar. 2017.

8. Spector, Dina. "Tiny Flying Robots Are Being Built To Pollinate Crops Instead Of Real Bees." Business Insider. Business Insider, 07 July 2014. Web. 15 Mar. 2017.