Prawn King Documentation

Rohan Chikkam

1 Introduction

- Agricultural Practice in Greater India
- Aquatic Agricultural Practice

2 Analysis

- Traditional Data Collection Methods
- Real World Problem
- Pre-development: Client Interviews
- Pre-development: Questionnaires
- Viable Solutions

3 Design

- The Solution
- Data Flow Model
- Project Pipeline

4 Implementation

- Overview1234
- Data Collection and Improvement

5 Testing

- Algorithmic Testing
- Reliability Testing
- Security / Data Integrity Testing
- Post-development: Client Interviews
- Post-development: Questionnaires

6 Evaluation

- Further Future Improvements
- Learning Outcomes
- Solution Effectiveness

7 Bibliography

- December 2019
- February 2020

8 Appendix

- Figures
- Source Code

1 Introduction

1.1 Agricultural Practice in Greater India

The majority of agricultural growth in South East India is that of cereal and traditional crop, consisting of rice, wheat, barley, millets and ragi among others. There are of course other practices as well... India is the world's largest producer of milk and pulses, ranking as the second largest producer of sugarcane, groundnut, vegetables, fruit and cotton. It is also one of the leading producers of spices, fish, poultry, livestock and plantation crops.[1]

It has held this position since 1951, but the agricultural sector has slowly been diminishing in contribution to the GDP ever since. One reason for this is because the yield per hectare of one of India's principal crops, rice, is 2177 kgs per hectare, lagging behind countries such as China and Brazil that have yield rates of 4263 kgs/hectare and 3265 kgs/hectare respectively.[2] This loss of competitive advantage has meant that the industry has become less profitable over time. Supported with the fact that the country has seen its employment slowly move from the secondary to tertiary sector, the number of people willing and with the ability to work in the sector has also diminished. Some business owners have looked into expanding their portfolio from the traditional produce for the area, aquatics being the main path being taken as of now.

1.2 Aquatic Agricultural Practice

I'll be focusing on the Crustacean based agriculture, namely Shrimp and Prawn, although many other types do exist; fisheries, algae, etc. The Aquatic Agricultural Sector is one that is extremely enticing for business owners looking to expand their venture, increase profits or for newly emerging capitalists - The main reason being, early profits. Many plots of aquatic agriculture are those that have been converted from previously traditional methods of farming, a way of doing this is to dig. Soil and land excavated from previous plots of land (which may have been used for crop, etc.), in preparation for tank conversion, can be sold for profit - good capital for the start of a business. The full production process consists of breeding with broodstock and spawners - producing eggs. However the majority of farmers' practice misses this part, it requires extremely high levels of specialist knowledge in the field, because of this most practice is led on from after this phase. Farmers buy 'spawn' or eggs, which then spawn into live.

Taking care of a spawn is a long and labour intensive process, each spawn lasting around 3 months, varying based of various factors and species. There are four types of farming possible; Traditional, Modified Extensive, Semi-intensive, and Intensive. The initial 3 methods make use of natural methods to capture spawn to grow and so are not widely used from a commercial stand point. Intensive methods require 24/7 attention, feeding with artificial diets , heavy aeration and water exchange to minimise environment deterioration.[3] However, are relatively the simplest and most profitable ventures to invest in.

On average, the tanks are fed 4 times a day, and even the smallest of businesses have to deal with more than 8-10 tanks at once (Any fewer and the business is not profitable). In between feed times, the intake of previous feed sessions are closely monitored using 'check-nets'. As well as this aerators are run on a strict timed schedule, if they are not run at the correct time the entire spawn dies. Even half an hour without required oxygenation and the consequences could be fatal. As well as this, pH and the size of the produce (count) are occasionally checked to make sure that the growth of the spawn is as projected, and profit margins will be met. All this is what it takes to run smoothly, on top of this we have to deal with disease, and calling advisors for medication, etc. Between a spawn completing growth and being retrieved for sale, and the next spawn being propagated, there is a 1 month turn around time where the tanks are cleaned and prepared. It is during this time that calculations are made for the financials of the previous spawn and projected financials for the next.

2 Analysis

2.1 Traditional Data Collection Methods

The Aquatic Agriculture sector is built up of organisations of multiple sizes, ranging from single owner plots to family run historic businesses to large multinational corporations. Regardless of the size of their operations the basis of production remains the same; Tanks are manned throughout the day and are fed by hand. Occasionally pH is measured to make sure the environment is maintained, similarly the size, weight, and health of the produce. Disease can also be of existence, risking the entire spawn if not treated appropriately. The difference between large company-owned and self-run plantations is that of financial capital, but more importantly of infrastructure and technology that they are given access to.

Traditionally, information collection has followed the given proceedings:

- Feed is distributed
- Amount of feed distributed is written on a sketchpad
- Repeat for every tank, every time feed is distributed in a day ***- Figure 1 (Appendix)***
- Watch 'check-nets' to monitor feed intake
- Amount of feed intake is written on a sketchpad
- Data is compiled when the time is found to do so
- Data may be digitised if required

Optional

- Check prawn health and size along with environmental conditions
- Contact advisors if necessary to treat tanks appropriately

2.2 Real World Problem

The fact is, a lot of man power is put into analysing data by hand, not due to a lack of technology but a lack of publicly available software. Large corporations have access to specialist tailored software which smaller businesses do not have access to, and cannot get financial support to get manufactured and engineered. Personally, I spent 2 months on the aquatic cultures learning the ins and outs of the process, I also took part in the calculations and congregation of the data at the end of the spawn's lifetime and felt as though the process could be helped from some modernisation.

Time and money can be saved given the appropriate tools to ease the production process. It is not the fact that a solution is not available or cannot be produced, it is the fact that this is an up and coming sector. Giving myself the unique opportunity to target a gap in the market before it is even formed because, inevitably, the Aquatic Agricultural Sector of Greater India is one that is on the rise, and will grow rapidly in market share in the coming years.

There are many possible solutions...

2.3 Pre-development: Client Interviews

My main point of contact for research and information regarding the topic, besides the internet, were my direct family. Many of whom are heavily involved in agriculture and even more who are familiar with Aquatic Agriculture. Clients are the most significant stakeholder when it comes to tackling the problem at hand, and so I aimed to make my research and analysis become based around people who come into direct contact with the problem on a day to day basis. Below are interviews of a few people and their opinion on the matter at hand:

2.4 Pre-development: Questionnaires

Since the businesses of concern are overseas and work on a different timezone it was not necessarily possible to contact all clients. As well as this, not all clients were willing or able to put aside time to take a telephone interview regarding the problem. To tackle this issue, and increase the feedback I could retrieve, I formed a questionnaire that could be passed around the network of agriculturalists that I tapped into. Responses are shown below:

2.5 Viable Solutions

- Solution 1 - Use of digital spreadsheet software on Personal Computers

  The use of spreadsheet software such as Microsoft's Excel has been put into use, this alleviates the tallying of data however makes it hard to analyse the data itself - while working with large dataset's, locally-hosted software may become very memory-intensive and many households do not have access to anything above the basic system requirements to even run the software let alone process these datasets. Furthermore, power-cuts and shortages are frequent in the region and downtime can last anywhere between a few hours to days, putting loss of data at the very top of the risks.

- Solution 2 - Hire an analyst and data handler

  Although analysis of data would be to a high standard and data handling would no longer be a problem on the business-owner's part, the expenses of hiring an expert in the field outweigh the benefits for many owners. Hence, why many do not choose to take this route - especially since labour in the region are paid on a daily rate.

  Neither of these solutions are widely in use due to the large capital investment required, only by large established businesses, who make up the minority of the Aquatic Agricultural Sector.

3 Design

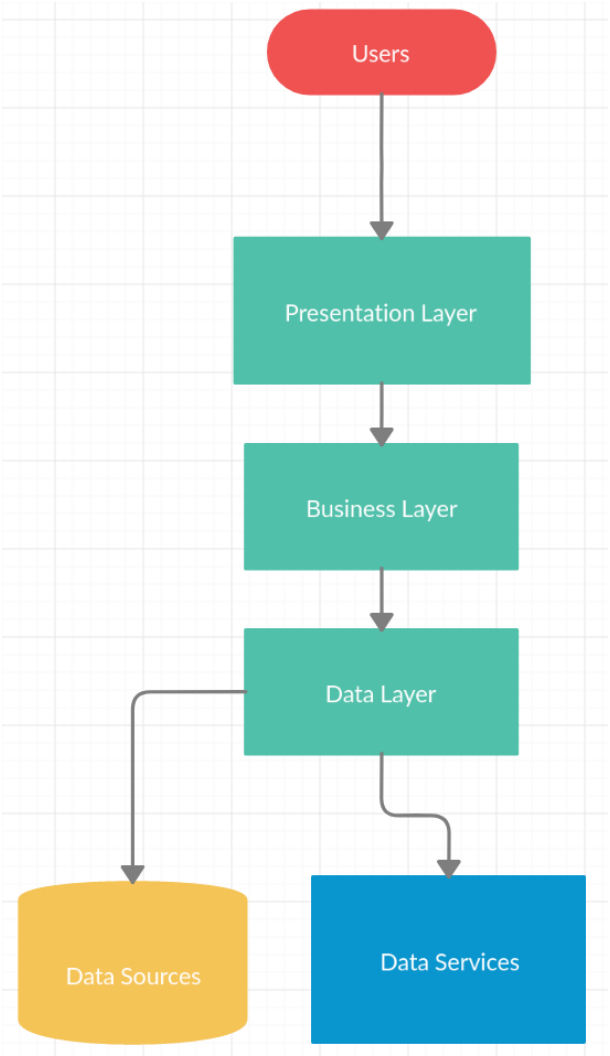
3.1 The solution

I aim to make an application that allows business owners and their labour to save data collaboratively. I want the data to be accessible from multiple locations and so will host the data in the cloud, as well as this I want all bulk operations to be carried out in the cloud to allow my application to run at the bare minimum system requirements. This also means in times of power outages, data loss is not a concern and access to the data will still be available on mobile devices. In order to enable this, I'll have to develop a desktop and mobile application, meaning I'd have to deal with cross-platform data communication, etc. I also aim to display data in an easy to use way and to use charts to provide useful analysis of the data itself. Overall, I'd like to keep burden on the host machine to a minimum.

To summarise:

- Multi-user access to data
- Remote storage and access to data
- All processing done in the cloud
- Cross platform client application
- Data analysis and data representation

3.2 Data Flow Model



By splitting the design and development process into these phases, it allows me to work with modularity - I can develop certain parts of the application without having to deal with problems of dependency or worrying about how they may work with other parts of the application.

It also means in future, I can update the presentation layer without having effected any of the backend processes, and vice versa - add and develop extra functionality on the backend while leaving the frontend fully functional and intact. After development is finished I can then present the changes to the user.

- **Presentation Layer** - Any application (PC/Android/iPhone) that allows the user to access and interface with their data.
- **Business Layer** - An intermediary layer to carry out processing and data coagulation, move processing away from host machines. (API Server, etc.)
- **Data Layer** - A layer that is responsible for storage and provision of data (SQL Server)
- **Data Sources** - The part of the Data Layer that is responsible for storing data (SQL DB)
- **Data Services** - The part of the Business Layer but built into the Data Layer that is responsible for processing of data (Organisation and analysis - SQL Server Services)

3.3 Project Pipeline

1) Create SQL Database (MySQL DB)
2) Create User Account (UA) Credentials DB
3) Create User Data DB
4) Create Stored Procedures: Create UA, Login, Data Retrieval, Updating, etc.
5) Hash passwords
6) Test DB Stored Procedures - SQL Injection Protection
7) Create RESTful API
8) Establish DB Connection from REST API
9) User Account Login over REST
10) User Account Creation over REST
11) Data Retrieval over REST
12) Test REST API - API Key/ Token Access only
13) Basic Test for REST API Procedures
14) Create WinForm
15) Working Log-in WinForm
16) Working Sign-Up WinForm
17) Data Representation
18) Data Editing/ Presentation
19) Create WebApp
20) Working Log-in WebForm
21) Working Sign-up WebForm
22) Data Representation
23) Data Editing/ Presentation
24) Create MobileApp
25) Working Log-in MobileForm
26) Working Sign-up MobileForm
27) Data Representation
28) Data Editing/ Presentation
29) Finalisation of UI/UX
30) Create Testing Container (Brute Force Environment)
31) Prepare Datasets
32) Bruteforce test REST API
33) Bruteforce test WinForm
34) Bruteforce test WebForm
35) Fix Test Errors + Retest
36) Prepare Hardcoded test cases
37) Release to public testing
38) Start Implementation of ML with Historical Data
39) Contact Advisors for expected output from Data, use data to train AI
40) Produce future predictions

4 Implementation

4.1 Overview

I will be using an iteration style development where I work through the project pipeline multiple times. I have outlined these are three main phases, Beta, Alpha and Public:

The **Beta** Phase is the most basic version of development and the initial phase that I will work on. It focuses on allowing me to become accustomed with the development software I will be using and will allow me to create a more refined version of my Project Pipeline for use in Alpha and Public Phases. Very little thought will be put into the visual aspect of the program, as most work and development will be on algorithmic structure and creating modularity in the system.

The **Alpha** Phase will follow the vague structure of the Beta Phase as well as any changes I've made to the Project Pipeline since. I will also start to mock-up some basic visuals for the End User and will begin to take in considerations like privacy and additional features to make my application more enticing.

The **Public** Phase may contain modules from the Alpha Phase especially with regards to the backend, a lot more focus will be put on the visual aspect of the application and I will start to develop mobile frameworks as well - since they can use the same back-end as the Desktop Application, I can use my visual mock-ups after translating them to mobile screens.

4.1.1 Setting up the SQL Server

I began to familiarise myself with the Microsoft Azure platform, however this was not viable for development as it was costing me almost £60 a month! I then planned to use a SQL Server instance running on my local machine. This would've been fine for local WinForm and WebApp development, however if I was to develop a mobile app or test my applications on other machines, I would not be able to do so. After a few google searches, I found a platform that allowed me to host free MySQL Server instances, Heroku with JawsDB. At release I cannot use this since the free tier plan has many constraints, but it is perfect for development since I will rarely be using more than one connection at a time. Full plan information can be seen in ***Figure 2 (Appendix)***.

4.1.2 A closer look at database structure

My Database structure is as follows:

- **Database** *pkdb*
  - **Table** *[dbo].prawnkingusers*
  - **Table** *[dbo].prawnkingdata*
  - **Table** *[dbo].prawnkingusersdatarel*
  - **SP** *ADDUSER*
  - **SP** *GETID*
  - **SP** *MKADMIN*
  - **SP** *UINFO*
  - **SP** *USEREXISTS*
  - **SP** *TACCESS*
  - **SP** *ADDTREF*
  - **SP** *ADDTANK*

I started using Stored Procedures as opposed to sending SQL Queries from the client for multiple reasons:

- Protects from Application Cracking - Even if the program becomes cracked or reverse engineered, the functions that can run on the SQL Server are limited as no instructions are passed from the client.
- Protects from SQL Server Attacks - Even if the SQL Requests are intercepted, they will not make sense to the intercepter, and even then only functions that already exist on the SQL Server can be run.

4.1.3 Creating the User Accounts table ([dbo].prawnkingusers)

Table Creation SQL Query:

```
CREATE TABLE `[dbo].prawnkingusers`
(
userID      INT auto_increment,
userName    VARCHAR(50)                NOT NULL,
userPass    VARCHAR(50)                NOT NULL,
FName       VARCHAR(50)                NULL,
LName       VARCHAR(50)                NULL,
PNumber     INT                        NULL,
isAdmin     BIT                        NULL,
CONSTRAINT `[dbo].prawnkingusers_userID_uindex` UNIQUE (userID),
CONSTRAINT `[dbo].prawnkingusers_userName_uindex` UNIQUE (userName),
PRIMARY KEY (userID)
);
```

The **first** field, userID, is created to be a both a Unique Key and a Primary Key. Using a Unique Key means that every user that gets added to the database has their own ID assigned to the. Using a Primary Key means that I can generate non-clustered indexes automatically as a new user is added to the database, simplifying the process, seen by *"auto_increment"*. I've also defined the field to be non-nullable, meaning that if a user exists, they must have an ID assigned to them. Having unique IDs will help me later on when I start to create relations between tables and user privellege management.

The **second** field, userName, is created to be unique, meaning no two users can have the same user name. The username is saved as cleartext.

The **third** field, userPass, has no constraints, other than of length 50 characters. The password is saved in a hashed format.

The **fourth, fifth and sixth** fields, FName, LName and PNumber, are user information, which will be encrypted using a key.

The **last** field, isAdmin, is a boolean containing whether or not the user is an administrator, being an administrator allows the user to carry out tasks with elevated permissions.

The use for this table is quite self explanatory, it is the table holding data responsible for all User Account Authentication and Privilege management, as well as all personal identifying data for the users.

4.1.4 Creating the User Data table ([dbo].prawnkingdata)

Table Creation SQL Query:

```
CREATE TABLE `[dbo].prawnkingdata`
(
tankID    INT auto_increment,
data      LONGTEXT    NULL,
comments LONGTEXT    NULL,
CONSTRAINT `[dbo].prawnkingdata_tankID_uindex` unique (tankID),
PRIMARY KEY (tankID)
);
```

The **first** field, tankID, is created to be a both a Unique Key and a Primary Key. Again for the same purposes as userID in the previous table

The **second and third** fields, data and comments, will contain the raw data collected from the user as well as any notes, etc that the user decides to save regarding the tank.

This table is responsible for holding the data of each location in a given user's portfolio, it'll be linked to multiple users in the user accounts table.

4.1.5 Creating the Relations table ([dbo].prawnkingusersdatarel)

Table Creation SQL Query:

```
1  CREATE TABLE `[dbo].prawnkingusersdatarel`
2  (
3  tankID    INT NOT NULL,
4  userID    INT NOT NULL,
5  FOREIGN KEY (tankID) REFERENCES `[dbo].prawnkingdata` (TANKID) ON DELETE RESTRICT ON
       UPDATE CASCADE,
6  FOREIGN KEY (userID) REFERENCES `[dbo].prawnkingusers` (userID) ON DELETE RESTRICT ON
       UPDATE CASCADE,
7  PRIMARY KEY (tankID,userID)
8  );
```

The **first** field, tankID, is created to be a Foreign Key which references the tankID column from the data table.

The **second** field, userID is created to be a Foreign Key which references the UserID column from the user information table.

This table contains the many-to-many relations between users and different tanks, since many users should be able to access the same tank, and many tanks are usable by the same user.

4.1.6 Creating the Add User SP (ADDUSER)

Stored Procedure Creation SQL Query:

```
1  CREATE PROCEDURE ADDUSER(
2  IN Usr varchar(50),
3  IN Pwd varchar(50),
4  OUT Rtn bit
5  )
6
7  BEGIN
8          IF EXISTS(SELECT userID FROM `[dbo].prawnkingusers` WHERE userName = Usr) THEN
9                  SET Rtn = 0;
10         ELSE
11                 INSERT INTO `[dbo].prawnkingusers` (userName,userPass)
12                 VALUES(Usr,Pwd);
13                 SET Rtn = 1;
14         END IF;
15 END;
```

This stored procedure adds new user if does not exist given Username and Password. **Lines 1-5** declare the input parameters (Username and Password to be added) and the output parameter (Rtn, success?)

Lines 8-14 is a conditional statement that checks whether a user with the provided username already exists; If it does, Rtn is set to 0 to signify a failure in account creation. Else, the account details are submitted into the table and a successful Rtn is passed.

Stored Procedure Execution SQL Query:

```
--username here
SET @Usr = '';
--password here
SET @Pwd = '';
CALL ADDUSER(
@Usr,
@Pwd,
@Rtn
);
--return here (1 - Success, 0 - Failure)
SELECT @Rt
```

4.1.7 Creating the Get ID SP (GETID)

Stored Procedure Creation SQL Query:

```
CREATE PROCEDURE GETID( IN Usr VARCHAR(50), IN Pwd VARCHAR(50), OUT Rtn INT )
BEGIN
SET Rtn = (SELECT userID FROM `[dbo].prawnkingusers` WHERE (userPass = Pwd AND userName
    = Usr));
END
```

This stored procedure returns the userID of the user with the provided username and password, given that these are of valid combination. **Line 1** declares the input parameters (Username and Password to be searched) and the output parameter (Rtn, ID?)

Line 3 Sets the result of the SELECT statement into the Rtn variable. The SELECT statement searches for the provided combination of credentials within the table and returns the respective userID. If no user exists, null is returned.

Stored Procedure Execution SQL Query:

```
--username here
SET @Usr = '';
--password here
SET @Pwd = '';
CALL getid(
@Usr,
@Pwd,
@Rtn
);
SELECT @Rtn
--return here (integer - Success, Null - Failure)
```

4.1.8 Creating the User Exists SP (USEREXISTS)

Stored Procedure Creation SQL Query:

```
CREATE PROCEDURE `USEREXISTS`( IN Usr VarChar(50), OUT Rtn INT )
BEGIN
IF (EXISTS (SELECT userID FROM `[dbo].prawnkingusers` WHERE (userName = Usr))) THEN
SET Rtn = 1;
ELSE
SET Rtn = 0;
END IF;
END
```

This stored procedure returns a boolean of whether a username is available for use or not given the username. **Line 1** declares the input parameters (Username to be searched) and the output parameter (Rtn, Available?)

Line 3 Sets the result of the EXISTS SELECT statement into the Rtn variable. The SELECT statement searches for the provided combination of credentials within the table and returns the

respective userID. If no user exists, null is returned. This existance is then converted into a Boolean with the EXISTS statement, this bool is then returned.

Stored Procedure Execution SQL Query:

```
--username here
SET @Usr = '';
CALL USEREXISTS(
@Usr,
@Rtn
);
SELECT @Rtn
--return here (1 - Exists, 0 - None)
```

4.1.9 Creating the Make Admin SP (MKADMIN)

Stored Procedure Creation SQL Query:

```
1  CREATE PROCEDURE `MKADMIN`(IN Usr varchar(50), IN Pwd varchar(50),IN UsrID int, OUT Rtn
       bit)
2  BEGIN
3  IF EXISTS (SELECT userID FROM `[dbo].prawnkingusers` WHERE userName = Usr AND userPass
       = Pwd AND `isAdmin?` = 1) THEN
4  IF EXISTS(SELECT userName FROM `[dbo].prawnkingusers` WHERE userID = UsrID) THEN
5  UPDATE `[dbo].prawnkingusers`
6  SET `isAdmin?` = 1
7  WHERE userID = UsrID;
8  SET Rtn = 1;
9  END IF;
10 ELSE
11 SET Rtn = 0;
12 END IF;
13 END;
```

This stored procedure attempts to promote the privelege of a user to admin given another set of user credentials who is already an admin. **Line 1** declares the input parameters (UserID of user to be promoted, and userName and userPass of the user who is carrying out the promotion) and the output parameter (Rtn, Success?)

Line 3 is a conditional statement that checks that the admin credentials provided are valid and that the user is actually an admin. If they aren't a failure is returned.

Line 4 is another conditional statement that checks that a user exists with the userID provided to be able to promote them to admin. Else returns failure.

Lines 5-8 edit the 'isAdmin?' value of the column of the user with the provided userID and sets the return value to 1 - success. This value is then returned.

Stored Procedure Execution SQL Query:

```
--username here
SET @Usr = '';
--password here
SET @Pwd = '';
--userID here
SET @UsrID = 0;
CALL mkadmin(
@Usr,
@Pwd,
@UsrID,
@Rtn
);
SELECT @Rtn;
```

```
14 --return here (1 - Exists, 0 - None)
```

4.1.10 Creating the User Info SP (UINFO)

Stored Procedure Creation SQL Query:

```
1  CREATE PROCEDURE `uinfo`(IN Usr varchar(50), IN Pwd varchar(50),IN fname varchar(50),
       IN lname varchar(50), IN pnumber varchar(50), OUT Rtn bit)
2  BEGIN
3  IF EXISTS (SELECT userID FROM `[dbo].prawnkingusers` WHERE userName = Usr AND userPass
       = Pwd) THEN
4  UPDATE `[dbo].prawnkingusers`
5  SET `FName` = fname,
6  `LName` = lname,
7  `PNumber` = pnumber
8  WHERE (userName = Usr AND userPass = Pwd);
9  SET Rtn = 1;
10 ELSE
11 SET Rtn = 0;
12 END IF;
13 END;
```

This stored procedure is used to add optional user information to the database, for future reference, contact, and personalisation options. **Line 1** declares the input parameters (UseruserName and userPass of the user who is carrying out the data addition, and the corresponding data, and the output parameter (Rtn, Success?)

Line 3 is a conditional statement that checks that the credentials provided are valid and that the user exists. If they aren't a failure is returned.

Line 4-8 update that given user's information

Line 9 returns a success code.

Stored Procedure Execution SQL Query:

```
1  --username here
2  set @Usr = '';
3  --password here
4  set @Pwd = '';
5  --first name here
6  set @fname = '';
7  --last name here
8  set @lname = '';
9  --phone number here
10 set @pnumber = '';
11 call uinfo(
12 @Usr,
13 @Pwd,
14 @fname,
15 @lname,
16 @pnumber,
17 @Rtn
18 );
19 select @Rtn;
20 --return here (1 - Exists, 0 - None)
```

4.1.11 Creating the Tank Access SP (UINFO)

Stored Procedure Creation SQL Query:

```
1  CREATE PROCEDURE `taccess`(IN UsrID int, OUT rtn varchar(255))
2  BEGIN
```

```
3 SET rtn =(SELECT GROUP_CONCAT(tankID) FROM `[dbo].prawnkingusersdatarel` WHERE userID=
      UsrID);
4 END;
```

This stored procedure is used to find the tanks that a user has access to, given the user ID. **Line 1** declares the input parameters (userID) of the user being queried, and the output parameter (Rtn, Success?)

Line 3 is a variable set statement that takes the output of the select statement and concatenates it into a single string delimited by commas, and so can be processed as a list of tanks for that user.

Stored Procedure Execution SQL Query:

```
1 --user id here
2 SET @UsrID = 0;
3 CALL taccess(
4 @UsrID,
5 @Rtn
6 );
7 SELECT @Rtn
8 --return here (Null - No tanks)
```

4.1.12 Creating the Add Tank SP (UINFO)

Stored Procedure Creation SQL Query:

```
1 CREATE PROCEDURE `addtank`(IN UsrID int, OUT rtn varchar(255))
2 BEGIN
3 IF EXISTS( SELECT userName FROM  `[dbo].prawnkingusers` WHERE userID = UsrID) THEN
4 INSERT INTO `[dbo].prawnkingdata`(data)
5 VALUES (null);
6 INSERT INTO  `[dbo].prawnkingusersdatarel`(tankID, userID) VALUES ((SELECT
      LAST_INSERT_ID()),UsrID);
7 SET rtn = 1;
8 ELSE
9 SET rtn = 0;
10 END IF;
11 END;
```

This stored procedure is used to add a new tank to the list of tanks and also create a relationship between the user who created it and that tank. **Line 1** declares the input parameters (userID) of the user being given a new tank, and the output parameter (Rtn, Success?)

Line 3 is a conditional statement that checks if the given userID exists in the user accounts db. If it doesn't error (0) is returned.

Lines 4 to 6 run if the conditional statement is successful, it adds a tank to the data db and then adds a relation between the user who created it and the id of the tank that was just created into the relations db.

Line 7 returns the success code.
Stored Procedure Execution SQL Query:

```
1 --userID here
2 SET @UsrID = ;
3 SET @Rtn = '';
4 CALL addtank(
5 @UsrID,
6 @Rtn
7 );
8 SELECT @Rtn
9 --return here (Null - No tanks)
```

4.1.13 Creating the Add Tank Relation SP (UINFO)

Stored Procedure Creation SQL Query:

```
1 CREATE PROCEDURE `addtref`(IN UsrID int, IN UsrAddID int, IN tnkID int, OUT rtn varchar
      (255))
2 BEGIN
3 IF EXISTS( SELECT userID FROM  `[dbo].prawnkingusersdatarel` WHERE userID = UsrID and
      tankID = tnkID ) THEN
4 INSERT INTO  `[dbo].prawnkingusersdatarel`(tankID, userID) VALUES (tnkID,UsrAddID);
5 SET rtn = 1;
6 ELSE
7 SET rtn = 0;
8 END IF;
9 END;
```

This stored procedure is used to add a new tank relation between a prexisiting tank and another user. **Line 1** declares the input parameters (userAddID) of the user being given a new tank, userID of a user already with access to the tank, and the output parameter (Rtn, Success?)

Line 3 is a conditional statement that checks if the given userID exists in the user accounts db against the provided tankID. If it doesn't error (0) is returned.

Lines 4 adds a new relational record into the relations db with the userID provided in (user-AddID) and the tankID

Line 8 returns the success code.
Stored Procedure Execution SQL Query:

```
1 --userID with access here
2 set @UsrID = ;
3 --userID to be given access here
4 set @UsrAddID = ;
5 --tankID to be given access to here
6 set @TankID =
7 set @Rtn = '';
8 call addtref(
9 @UsrID,
10 @UsrAddID,
11 @TankID,
12 @Rtn
13 );
14 SELECT @Rtn
15 --return here (Null - No tanks)
```

5 Bibliography

5.1 December 2019:

[4]Ravindra Chikkam - Personal Data Collection Measures

5.2 February 2020:

[1]FAO - www.fao.org - "India at a glance"
[2]BASF - www.basf.com - "How sustainable farming in India can secure its food for the future"
[3]The Fish Site - www.thefishsite.com - "How to Farm Indian White Prawn"

6 Appendix

6.1 Figures

		Tank 1							Tank 2						
Day	Date	1	2	3	4	Total	COUNT	Growth	1	2	3	4	Total	Count	Growth
1	17/07/2019				1	1	20	50				1	1		
2	18/07/2019				2	2						2	2		
3	19/07/2019				2	2						2	2		
4	20/07/2019				3	3						3	3		
5	21/07/2019				4	4						4	4		
6	22/07/2019				4	4						4	4		
7	23/07/2019				4.5	4.5						4.5	4.5		
8	24/07/2019				4.5	4.5						4.5	4.5		
9	25/07/2019				5	5						5	5		
10	26/07/2019				5	5						5	5		
11	27/07/2019				5.5	5.5						5.5	5.5		
12	28/07/2019				6	6						6	6		
13	29/07/2019				6.5	6.5						6.5	6.5		
14	30/07/2019				7	7						7	7		
15	31/07/2019	1.9	1.9	1.9	1.9	7.6			1.9	1.9	1.9	1.9	7.6		
16	01/08/2019	2	2	2	2	8			2	2	2	2	8		
17	02/08/2019	2.25	2.25	2.3	2.3	9			2.3	2.3	2.3	2.3	9.2		
18	03/08/2019	2	2	2	2	8			2.5	2.5	2.5	2.5	10		
19	04/08/2019	2	2	2	2	8			2.7	2.7	2.7	2.7	10.8		
20	05/08/2019	2	2	2	2	8			2.9	2.9	2.9	2.9	11.6		
21	06/08/2019	2.5	2.5	2.3	2.5	9.75			4.5	4.5	4.5	4.5	18		
22	07/08/2019	2.5	2.5	2.5	2.5	10			4.5	4.5	4.5	4.5	18		
23	08/08/2019	2.5	2.5	2.5	2.5	10			4.5	4.5	4.5	4.5	18		
24	09/08/2019	3	3	3	3	12			5	5	5	5	20		
25	10/08/2019	3	3	3	3	12			5	5	5	5	20		
26	11/08/2019	2.5	2.5	2.5	2.5	10			5	5	5	5	20		
27	12/08/2019	2.5	2.5	2.5	2.5	10			5	5	5	5	20		
28	13/08/2019	2.5	2.5	2.5	2.5	10			5	5	5	5	20		
29	14/08/2019	3	3	3	3	12			5	5	5	5	20		
30	15/08/2019	3	3	3	3	12			6	6	6	6	24		

Figure 1: *An extract of an eighth of a month's data collected by a small scale aquatics owner.*[4]

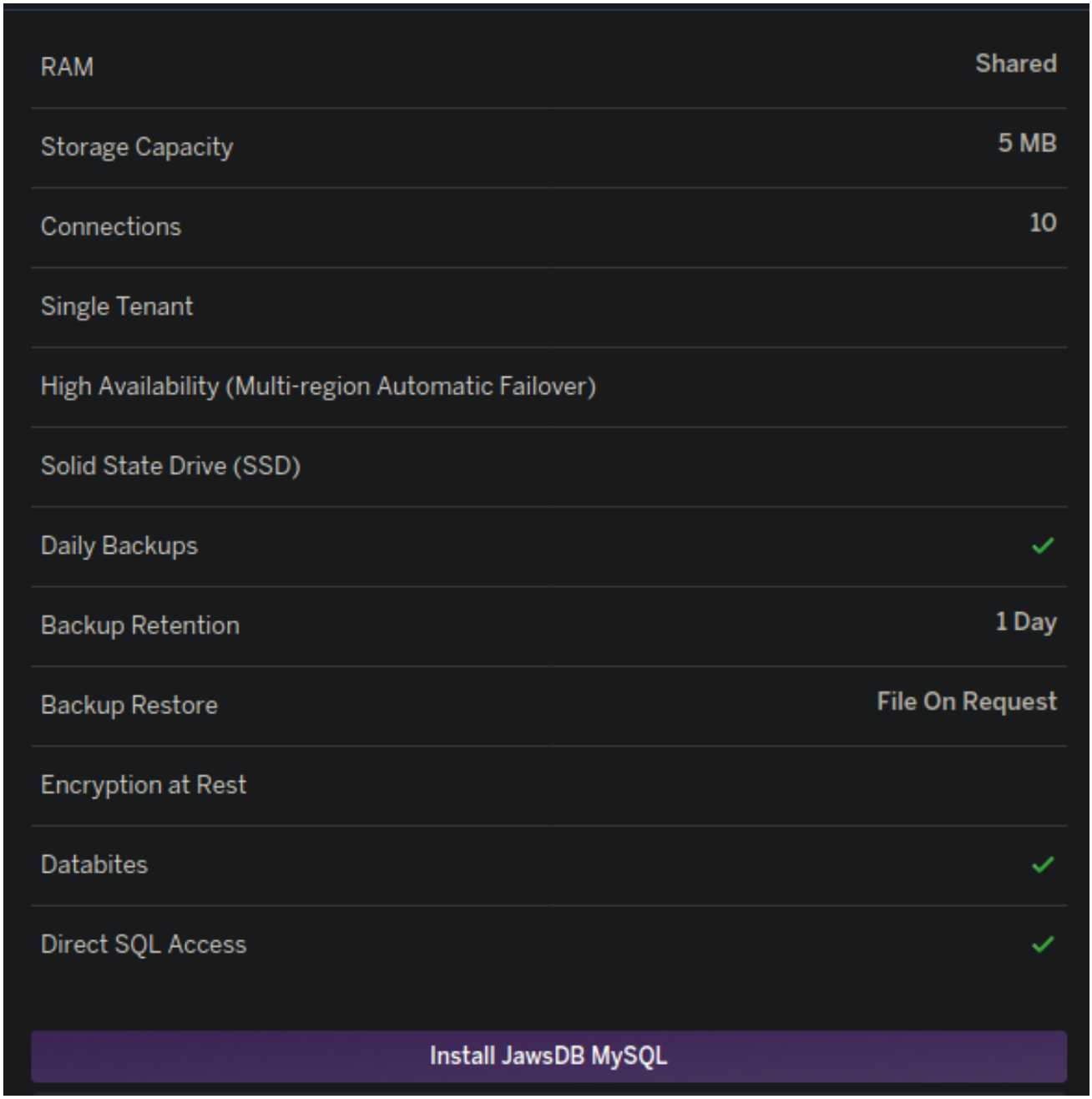


Figure 2: *The Free Tier Plan from heroku hosting the JawsDB add-on.*

6.2 Source Code