

**Festival Control System.**

**Final Report.**

Client: Mister George

Service Provider: Step-Soft

Members: Atanas Naydenov, Dimitar Markov, Dmitrii Orlov, Hristian Vasilev.

Content

[Description of Deliverables: 4](#_Toc454863297)

[Website 4](#_Toc454863298)

[Technologies used 4](#_Toc454863299)

[Website file system organization 4](#_Toc454863300)

[Functionality 5](#_Toc454863301)

[Page description 7](#_Toc454863302)

[DATABASE 10](#_Toc454863303)

[Design 10](#_Toc454863304)

[Tables 10](#_Toc454863305)

[Views 12](#_Toc454863306)

[Events 13](#_Toc454863307)

[Applications 14](#_Toc454863308)

[Bracelet management 14](#_Toc454863309)

[Visitor support 15](#_Toc454863310)

[Statistics app *“StatsApp”* 17](#_Toc454863311)

[Shopping application 24](#_Toc454863312)

[Camping Entry and Exit 26](#_Toc454863313)

[Event Entry and Exit 27](#_Toc454863314)

[Loaning Item Application 27](#_Toc454863315)

[Activity entrance application 29](#_Toc454863316)

[Activity exit application 30](#_Toc454863317)

[Project Plan: 30](#_Toc454863318)

[Overview: 30](#_Toc454863319)

[Formal client: 31](#_Toc454863320)

[Project Team: 31](#_Toc454863321)

[Current situation: 32](#_Toc454863322)

[Project justification: 32](#_Toc454863323)

[Problem description: 32](#_Toc454863324)

[Project goal: 32](#_Toc454863325)

[Deliverables and non-deliverables: 33](#_Toc454863326)

[Deliverables: 33](#_Toc454863327)

[Non-Deliverables: 33](#_Toc454863328)

[Constraints: 34](#_Toc454863329)

[Risks: 34](#_Toc454863330)

[Project phasing: 36](#_Toc454863331)

[Phase 1: 37](#_Toc454863332)

[Phase 2: 38](#_Toc454863333)

[Work Division Table: 39](#_Toc454863334)

[Personal Evaluation: 40](#_Toc454863335)

[Atanas: 40](#_Toc454863336)

[Dimitar: 40](#_Toc454863337)

[Dmitry: 41](#_Toc454863338)

[Hristiyan: 42](#_Toc454863339)

# Description of Deliverables:

## Website

|  |  |  |
| --- | --- | --- |
| Testing data: | Tony Montana | Jennifer Burges |
| Email | [tm@gmail.com](mailto:tm@gmail.com) | [jb@gmail.com](mailto:jb@gmail.com) |
| Password | tnmntna | jnfr |
| Ticket | Yes | No |
| Tent | None | None |
| Activities | X2 reserved | X1 reserved |
| Items | X3 loaned | X0 loaned |

### Technologies used

Our team used different IDEs, with the mainly used being:

* Atom
* Netbeans
* Sublime
* Visual Studio Code

Our web application uses the following technologies

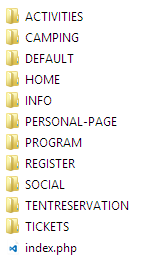
* PHP for the back-end of the website
* MySQL for the database
* CSS and JavaScript for the front-end

We also used some libraries and frameworks like:

* jQuery - We used because of the easier way of implementing animations and interacting with the DOM tree
* Bootstrap – Mainly we made use of the grid that the framework gives
* Sweet Alerts - Used for displaying nice looking messages (examples of use at Personal Page, Registration/Login page, and others)  
  http://t4t5.github.io/sweetalert/
* Alertifyjs – Used for displaying nice looking log notifications. Used at the Activities page and also in the framework to display the terms and conditions.  
  http://alertifyjs.com/

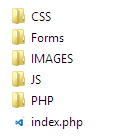
### Website file system organization

Our group chose this structure because it best suited the purpose of the application.  
The root folder has the following structure:



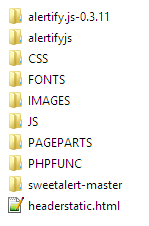
* It is divided into 11 subfolders and an ‘index’ file.
* The starting point of the application is the file ‘index.php’.
* The website directoroty is organised in such a way that the files can be easily accessed by the main file.
* Every subfolder contains data about one particular page.
* Every page folder which also contain a starting index file and subfolders with folder for every particular thing? that is used.
* The folder called ‘DEFAULT’ serves to store files that are shared between all the pages. There can be found images, header, footer, some default styles and scripts, libraries fonts and etc

Each page folder is then separated into subfolders for the page specific media, scripts, PHP code and other data.

This is an example taken from the activities page directory, other separate pages follow the same/similar structure:

* CSS folder stores all the page specific stylings
* Forms – stores all the additional forms and templates that are loaded in the page by AJAX during runtime
* Images – stores the media and images that are used in this page
* JS – stores the JavaScript code for the page
* PHP – stores the PHP codes for the page to process the server queries and interaction

The “DEFAULT” directory is used to store shared media, codes, frameworks and styles. A lot of references, requires and reuses in the pages refer to this folder’s contents.

* It has 9 subfolders
* Aertify.js-0.3.11 – is a folder that stores the framework
* Alertifyjs – is a folder storing a slightly different version of the Alertifyjs framework
* CSS – a directory storing the shared styles for every page  
  e.g: footer.css, integral.css
* Fonts – a directory that stores the fonts that are used on the website. These are Bauhaus93 and Odin Rounded (<http://www.1001fonts.com/odin-rounded-font.html>)
* Images – a directory for the common images. It includes the images that are used in the footer, logos, social network logos and others.
* Pageparts – a directory that stores the reused/shared parts of the pages: Footer.php, FooterLinks.php (which are nested inside of the Footer.php), FooterLinksHome.php – the part that is used in the home page (root/index.php), since it’s located on a higher level than other pages
* PHPFUNC – a directory that stores the serverside codes that are used at every page, e.g Logout.php (since the logout is accessible in the header navbar at every page)
* Sweetalert-master

### Functionality

#### Overview

For every page, except the ‘Home page’, we use the same menu and footer, which are loaded with PHP from the ‘DEFAULT’ folder. Every page shares the styling of those elements from the DEFAULT/CSS directory as well.

The header’s last link – account does not only serve as a redirect mechanism to either login/register page or personal page, but also provides the way to log out on the mouse hover.  
The footer functionality as follows: *“news & social”* redirects to a respective page, *“contact us”* opens a new window for a default client’s mailing provider with a brief template for an email to [jobsdoneuniversal@gmail.com](mailto:jobsdoneuniversal@gmail.com) (this is a valid email address), *“terms and conditions”* prompts with a message box with the terms and conditions – made using alertifyjs framework. This functionality is available at every page

#### JS and PHP

What is specific is that all our PHP scripts that are invoked on some button click, are executed via AJAX which means that to any button of that kind(e.g. register, login) is attached a click event which calls a JavaScript file who executes the PHP script, processes it and retunes a relevant response asynchronously.

While developing the codes for the website (both JS and PHP) the quality of them and their patterns were changed gradually as more good practice conventions were learned. The PHP code is very far from perfect. It uses pure PHP, and does not implement any frameworks or additional libraries, which makes it a) overly complicated; b) the separation of logic and visualization is not possible using such way. In the AJAX-PHP interaction pattern the server responses are categorized by the unique messages that are echoed from in the PHP code upon the termination of a certain method. However, for all the server and MySQL database interaction the object oriented PDO pattern was used.

E.G from the login js-php interaction at the REGISTER/LOGIN page:

|  |  |
| --- | --- |
| *// from the js/auth/ajax-login.js*  *…*  success: function(response) {  if (response.**includes('logged'**)) {  …..  } else {  ……  }  } | *// from the php/login.php*  *….*  $stmt->execute(array(":email"=>$user\_email));  $row = $stmt->fetch(PDO::FETCH\_ASSOC);  // $count = $stmt->rowCount();  if($row['PASSWORD']==$password && $password !="" ){  **echo "ok-you logged in"**.print\_r ($row);  $\_SESSION["USER\_ID"] = $row['USER\_ID'];  }  else{  echo "email or password does not exist.";  }  … |

PHP code was used to display the user/condition relevant data on the webpages. It was used to populate the contents of the Personal Page modules, create the forms to reserve activities, show the user reserved activities and so on. Example code provided below:

*// from the ACTIVITIES page index.php*<?php

for ($i = 0; $i < count($activities); $i++) {  
 echo '  
 <li class="act\_item">  
 …   
 <!--<form action="php/processreservation.php" method="post">-->

<form class="formreservation">

<div class="activity\_info">

<p class="text actname">' . $activities[$i] . /\* the name 0 7 \*/'</p>

…  
 Places available: ' . $activities[$i+=1] . /\* places 4 \*/'

…  
 <input type="checkbox" class="chkbox" required   
 name="reserve" checked="true" value="yes">  
 <label class="chkbox"> I want to reserve </label>

…

<button type="submit" class="btn text resbtntext">RESERVE</button>

…  
 </li>';

} ?>

As for the JS codes, a lot of JQuery code was written to interact with the DOM. It made it easy to animate the webpages and to work with the ‘static’ and user-input data. E.G: a lot of JQuery code was used in the TENT RESERVATION page.

#### APIs

For this website the following APIs were used:

* Twitter API   
  In the ‘SOCIAL and NEWS’ page reached from the footer, the twitter provided APIs are used both to present a twitter news feed and also to share the page via a ‘tweet’ by pressing the according button.
* Facebook API  
  In the ‘SOCIAL and NEWS’ page the ‘share’ button is implemented to post directly to the Facebook timeline.
* Google+ API  
  In the ‘SOCIAL and NEWS’ page the ‘share’ button is implemented to post directly to the Google+ timeline.
* Google Maps API  
  In the INFO -> LOCATION page the Google maps API was used to show display the event place and facilitate locating the venue by providing an easy way to set a route.

### Page description

#### Activities page

This page is used to view the activities that can be attended during the festival. The registered users can book a place at the activity and cancel it at any moment. This ‘web application’ replaces the activity reservation windows application that was declared in the project plan. This page is more reasonable and user-friendly than a C# windows application with the same functionality.

The PHP and JavaScript are very faulty and is not implemented to run effectively.   
The code specifications on this page:

* JQuery to animate the activity boxes
* Ajax – to – PHP communication pattern to process reservations and cancelations
* On-page PHP to retrieve the activities from the database
* On-page PHP to display the user reserved activities, if he is registered, or a message to login if he is not.
* Alerifyjs notifications to display logs in the left lower corner of the screen about successful/unsuccessful operations.

#### Camping page

The page consist mostly of static information and a dynamic PHP script for posting a link to the tent reservation for several conditions: a) more than one camping spot available; b) only one spot available; c) no places are left. This code places an image-link by pressing which the visitor is redirected to the tent-reservation page.

a) b) c) 

#### Home and Info pages

Consist of static information and information about the event.   
The home page is specific as at first, a large banner appears at the top part of the window and the navigation bar is below it. Once the page is scrolled down, the navigation bar is lifted to the top of the window and fixed. For this, the JQuery code was created that changes the class of a navbar so it would fit the appropriate styling. Below is the snippet of the script from the index.php.

var logo = document.getElementById('logo');  
 var potato = document.getElementById("potato");  
 $(document).on("scroll", function() {  
 if ($(this).scrollTop() > 580) {  
 $("#header").removeClass("headernotscrolled");  
 …  
 logo.src = "DEFAULT/IMAGES/logohead.png";  
 potato.style.display = "inline";

} else {  
 $("#header").removeClass("scrolled");  
 ….  
 potato.style.display = "none";  
 }  
 });  
 </script>

#### Personal page

There the user can find all his personal information he has provided at the register page, information about his camping status, finances and etc. It consists of dynamic and static parts. The information is separated into separate modules such as “*General*” – for the general user data such as name, address, email; *“Balance”* – used to manage the visitors balance; *“Bracelet”* – used for viewing the visitor’s ticket payment and bracelet information; *“Your Tipi”* – shows the data (number and the tent leader) about the camping spot for which the visitor is registered; *“Items”* – displaying the information about the items which the visitor had loaned; “Activities” – showing the list of the activities.



The buttons for hiding and showing are implemented with jQuery and some easy showing and hiding html elements, manipulating styles and loading content with AJAX. Every button has an attached click event, which executes a corresponding script.

Every time the user accesses the page, the system executes few PHP scripts. Via these PHP scripts are made queries to the database and the information about the particular user is retrieved. Whenever the user wants to update his balance, he can go to the balance field and fill the desired amount to be added to his account. Considering the PHP scripts they are still not very optimized in terms of security and quality of the code (a lot of repetitive code).

#### Program page

It consists of static information about the program of the event. There you can see a few tabs whose content is dynamically loaded with jQuery and AJAX. It also features some animations.

#### Register page

It has 2 tabs one of them is for registering the other is for logging in. The content is loaded dynamically with the help of jQuery and Ajax. On the page, there is also some HTML5 and JavaScript validation of inputs, e.g. the email entered is forced to be the same as the confirm email (same applies for the password).

If we look at the PHP script and the way the whole registering and logging in works, we will see as mentioned in the overview that the scripts are executed via AJAX code that is executed by clicking on the corresponding button. Considering the PHP scripts, they are still not very optimized in terms of security and the code quality (a lot of repetitive code).

#### Social page

There the users can find a twitter feed and the share buttons for Facebook, Twitter and Google+. Everything is implemented by using the respective APIs.

#### Tent reservation page

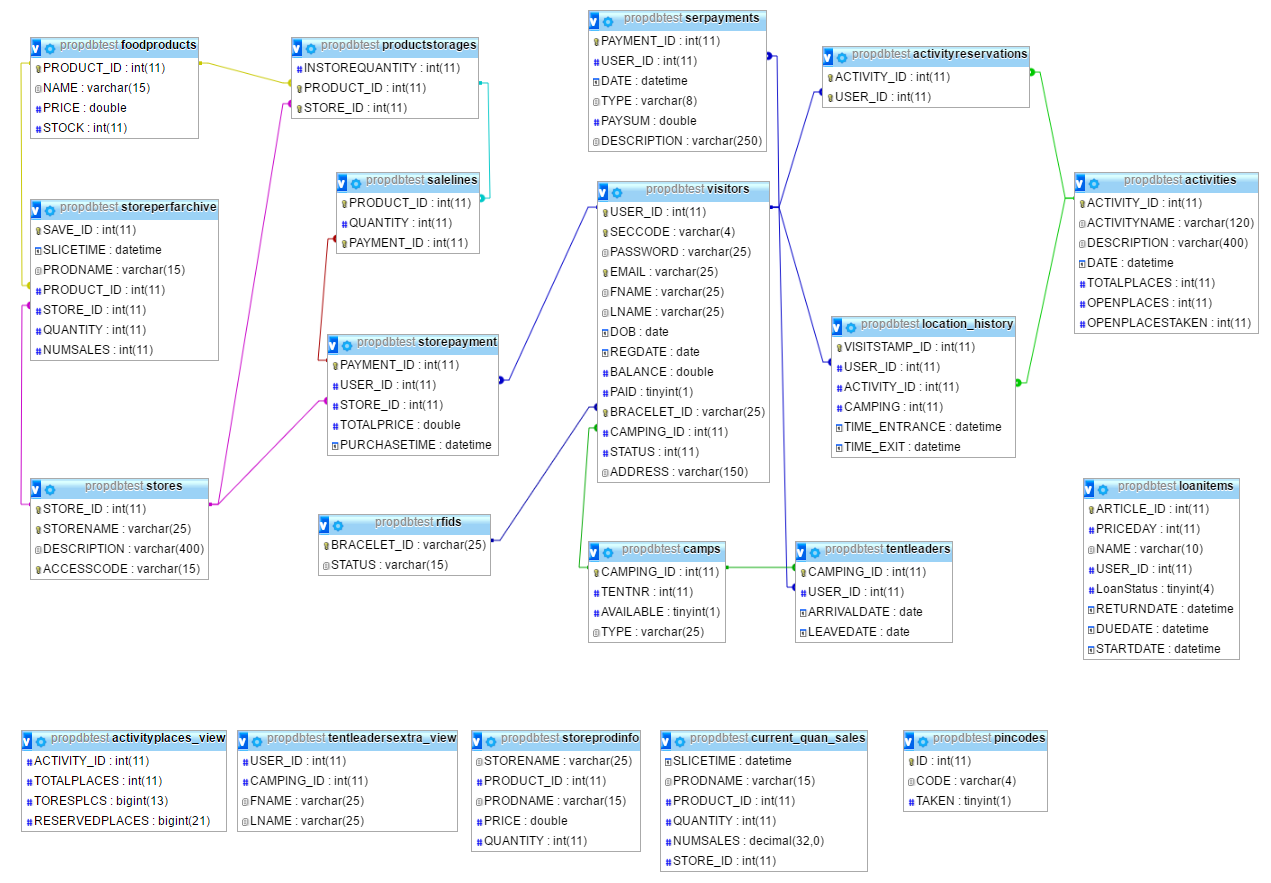
At this page the camping spots can be reserved for a group of up to 6 people. One user is required to pay the whole sum for the camping from his balance and also will be the responsible person for the camping spot. He has to enter his credentials once again to ensure the secure user authentication.   
  
The codes that are used to generate the input fields are written using JQuery and pure Javascript. The server interaction is done via the AJAX and PHP.

Tickets page   
This page is used to purchase the tickets for the events – this is a required condition for obtaining the bracelet and reserving a tent at the entrance. The user is redirected to this page right after the registration.

There are three types of tickets, which generally differ only in price and have do not affect the general user experience in any way (Our team supposes that is how all the festivals work). To purchase a ticket the visitor has to enter his credentials to ensure his validity. For the purchase a new window is loaded in the center of the page.

\*For further details refer to the code and comments.

## C:\Users\Dmitry\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Screenshot (137).pngDATABASE



Design

The ERD diagram above depicts a complete design of the database.

It features 16 tables, 4 views, and 1 event. Those are required for different interactions with the database via the applications and the website. The 5 views were created in order to facilitate the data retrieval from the database. 1 event that is not present in the ERD diagram is called SalesSlicerEvent and is executed every hour in order to save the ‘slices’ of the quantity and sales of the products per each shop.

### Tables

* Activities.  
  The table to keep the data about the events. Includes the name, id, date, description, number of places (of three types – open/take/total)
* ActivityReservations.  
  This is an intersection table to store the data about the visitor’s reserved activities.
* Camps.  
  This is a table to store the data about the camps. It has the attributes to store the availability of a camp, its ID, number and type (which is not used for anything now).
* Foodproducts.  
  This is a table to store the data about the food products that are sold at the stores. It serves as a general warehouse – from there all the products are divided between the shops.
* Loanitems.  
  This is a table to have all the data about the items that can be loaned during the event. This table is not connected with a relation in the ERD, but in fact the USER\_ID attribute references to the USER\_ID of the visitors. If the item is loaned the USER\_ID is retrieved from the visitor and if not it is set to 0. The dates signify the dates when the item was returned, when it is due to be returned and when it has been taken. – when a new visitor starts using the item the data is overwritten.
* Location\_History.   
  This table is used to store the location of the visitors – it was designed to secure the entrances and exits to the events and camps from fraud. It has the attributes: activity\_id and camping, which are set former to the activity\_id of the one the visitor is entering and camping is set to 1 if the visitor is entering the camping. Potentially this table can also be used to review the statistical information about the popularity of the activities and density of the camping.
* Pincodes.   
  This table was created to store the unique 4 char that were supposed to be used as an additional mean to identify the visitor. However, at the current moment this table is not used anyhow.
* ProductStorages.  
  This association table is used to see the quantity of the products present in the stores.
* RFIDS.  
  This data keeps the data about the bracelets – IDs and statuses. A range constraint applies to the statuses: they can be any of only three following values: STAND\_BY, ACTIVE, DEACTIVATED.
* SaleLines.  
  This is an association table used to keep track of the products that were bought at one payment. Thus the foreign key Payment\_ID references to the StorePayment table. The data about the products is obtained from the reference to the FoodProducts table.  
  This follows a concept of a following data design:

A person could have made several payments in different stores for different sets of products

A sale-line stores a single product and its quantity purchased that was included in the payment

A payment can have many products in one check

* SerPayments.  
  This table stores the information about all “service” payments performed by the users. The payments for the tickets and tents are considered to be service payments.
* StorePayment.  
  \*This table is the only table that does not follow the general naming convention due to a typo on an early stage and afterwards too many code was written using the faulty name.  
  This association table stores the payments in the stores of the visitors. It has a Payment\_ID, PurchaseTime – which shows the time when the purchase was made –, a Store\_ID to store the reference to a store where the payment was made, a TotalPrice – to store the complete amount that was paid – and a User\_ID to store the reference to which Visitor has performed this payment.
* StorePerfArchive.  
  \*This table has a different meaning and thus doesn’t follow the naming convention.  
  This table is populated via the SalesSlicerEvent event every hour. It stores the number of sales of a certain product, its name, id, current (at the time of insert) quantity in a store for every store, and also stores a time when the save (‘slice’) was done and a store\_id to keep track of the stores.
* Stores.  
  This table keeps the store data, with its name, ID, description and accesscode. The former two attributes are not used in any applications, but are valuable for further development in case of continuing production. The access code can be used to restrict the use of certain applications, and the description - to display the information on the website for the visitors.
* Tentleaders.  
  This table keeps the data about the camping reservations – the name is slightly updated and doesn’t comply correctly with the current purpose. It stores the data about the user who had paid for the tent group – thus the responsible person (chief), camping\_id, leave date and arrival date – which signify for which dates the camp is going to be reserved.
* Visitors.  
  *“The table of all tables”*  
  This table stores all the data about a certain visitor – all his/her general information provided at the registration, the camping and bracelet associated with the visitor. The status can be either 1/0 (Tinybit or a Boolean) and represents whether a visitor has entered the event or not. The paid attribute shows whether the user has purchased a ticket. Passwords now are stored in an open way, which is a very bad practice, but we decided to leave it this way for the time being.

### Views

* ActivityPlaces\_view.  
  This view is used to calculate the places for the events – the total number of places/2 is the number of places available to be reserved. Reserved places – the number of visitors registered for an event via the website. The number of open places at the moment of the event start = totalplaces – reservedplaces.  
  This view is defined as:   
    
  SELECT `a`.`ACTIVITY\_ID` AS `ACTIVITY\_ID`, `a`.`TOTALPLACES` AS `TOTALPLACES`, floor((`a`.`TOTALPLACES` / 2)) AS `TORESPLCS`, count(`ar`.`USER\_ID`) AS `RESERVEDPLACES`   
  FROM (`propdbtest`.`activities` `a`   
   JOIN `propdbtest`.`activityreservations` `ar`   
   on((`a`.`ACTIVITY\_ID` = `ar`.`ACTIVITY\_ID`)))   
  GROUP BY `a`.`ACTIVITY\_ID`
* Current\_quan\_view.  
  This view is used in pair with a storeperfarchive and follows the same structure but instead of being populate each hour, stores the data of a current moment. This is used in the statistics to compare historic data and the current.  
  It is defined as:   
    
  SELECT now() AS `SLICETIME`,`fp`.`NAME` AS `PRODNAME`,`ps`.`PRODUCT\_ID` AS `PRODUCT\_ID`,`ps`.`INSTOREQUANTITY` AS `QUANTITY`,sum(`sl`.`QUANTITY`) AS `NUMSALES`,`ps`.`STORE\_ID` AS `STORE\_ID`   
  FROM (((`propdbtest`.`foodproducts` `fp`   
   join `propdbtest`.`productstorages` `ps`   
   on((`fp`.`PRODUCT\_ID` = `ps`.`PRODUCT\_ID`)))   
   left join `propdbtest`.`storepayment` `sp`   
   on((`ps`.`STORE\_ID` = `sp`.`STORE\_ID`)))   
   join `propdbtest`.`salelines` `sl`   
   on(((`sp`.`PAYMENT\_ID` = `sl`.`PAYMENT\_ID`)   
   and (`sl`.`PRODUCT\_ID` = `ps`.`PRODUCT\_ID`))))   
  group by `ps`.`PRODUCT\_ID`,`ps`.`STORE\_ID`;
* Storeprodinfo.  
  This view is used to facilitate the access to the information about the products currently located in the store.  
  It is defined as:  
    
  SELECT `st`.`STORENAME` AS `STORENAME`,`ps`.`PRODUCT\_ID` AS `PRODUCT\_ID`,`fp`.`NAME` AS `PRODNAME`,`fp`.`PRICE` AS `PRICE`,`ps`.`INSTOREQUANTITY` AS `QUANTITY`   
  FROM ((`propdbtest`.`productstorages` `ps`   
   join `propdbtest`.`foodproducts` `fp`   
   on((`ps`.`PRODUCT\_ID` = `fp`.`PRODUCT\_ID`)))   
   join `propdbtest`.`stores` `st`   
   on((`st`.`STORE\_ID` = `ps`.`STORE\_ID`)))   
  order by `fp`.`NAME`
* Tentleadersextra\_view.  
  This view is used to facilitate the access to the information about the tent leaders. It retrieves the visitor data based on the user\_id of a leader.  
  It is defined as:   
    
  SELECT `t`.`USER\_ID` AS `USER\_ID`,`t`.`CAMPING\_ID` AS `CAMPING\_ID`,`v`.`FNAME` AS `FNAME`,`v`.`LNAME` AS `LNAME`   
  FROM (`propdbtest`.`tentleaders` `t`   
   join `propdbtest`.`visitors` `v`   
   on((`t`.`USER\_ID` = `v`.`USER\_ID`)));

### Events

* SalesSlicerEvent.  
  This event is used to make hourly ‘saves’ or ‘snapshots’ or ‘slices’ of the product statistical data per shop. It inserts the data into the StorePerfArchive table on every interval. This table is described above.  
  The event’s function is defined as:   
    
  INSERT INTO storeperfarchive (SLICETIME, PRODNAME, PRODUCT\_ID, QUANTITY, NUMSALES, STORE\_ID) SELECT NOW(), fp.NAME, ps.PRODUCT\_ID, ps.INSTOREQUANTITY, SUM(sl.QUANTITY) AS NUMSOLD, ps.STORE\_ID   
   FROM foodproducts fp  
   join productstorages ps   
   on fp.PRODUCT\_ID = ps.PRODUCT\_ID  
   left outer join storepayment sp   
   on ps.STORE\_ID=sp.STORE\_ID   
   join salelines sl   
   on sp.PAYMENT\_ID = sl.PAYMENT\_ID and sl.PRODUCT\_ID = ps.PRODUCT\_ID   
   GROUP BY ps.PRODUCT\_ID, ps.STORE\_ID

This database design proved itself usable and informative enough despite its drawbacks and imperfections. The final design differs a lot from the one declared initially. It does not implement some concepts of the initial idea, but improved a lot in other spheres.

## Applications

This section describes the application covering their details and functionality overviews.

Some of the applications have the class AutoClosingMessageBox implemented. That is a class that upgrades the message box to take one extra parameter of type integer. This extra parameter serves as a timeout after which the message box automatically closes.  
  
The following applies to all the applications:

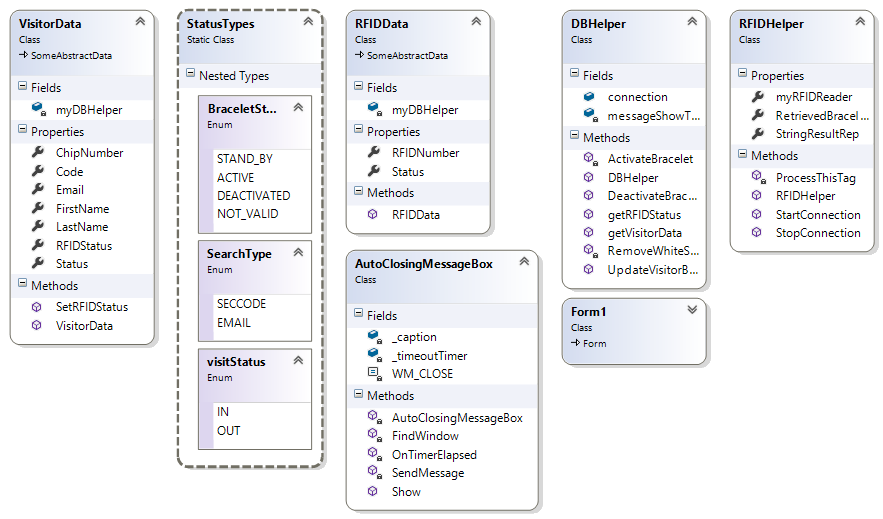
* The deployment is not performed
* The database connection settings are stored in a respective DLL – DBConnectionDLL – and are shared by the applications.

### Bracelet management

#### Overview

This application is to be used in the support/bracelet management booths. It's main functionality is to manage the bracelets: assign, issue, deactivate and change the bracelets for certain visitors. It is built upon two communication channels: PC-MySQL database and PC-RFID reader.   
This application should be used to issue the bracelets for the new arriving visitors. Example testing data: email – [abc@abc.abc](mailto:abc@abc.abc) / [jf@jf.jf](mailto:jf@jf.jf) / seccode - KARL

#### Class diagram



* Visitor Info:   
  Visitor information retrieved for the purposes of this application complies with the database visitor table and uses the following attributes: EMAIL, FNAME, LNAME, SECCODE, BRACELET\_ID, STATUS.  
  The information is obtained by searching for a visitor via email/secret code input. The following method is used to retrieve the respective data:   
    
   public VisitorData getVisitorData(StatusTypes.SearchType searchAttribute, string whereClauseValue)   
  In the body of this method the following query is executed:   
   String sql = "SELECT EMAIL, FNAME, LNAME, SECCODE, BRACELET\_ID, STATUS FROM VISITORS WHERE UPPER(" + whereClauseAttribute + ") =" + " UPPER(\"" + whereClauseValue + "\")";
* RFID data:   
  Complies with the database structure as well. Obtained from the database via the visitor information or from the RFID reader.
* To interact with the RFID reader and the Database the according RFID helper and DB helper classes were used.

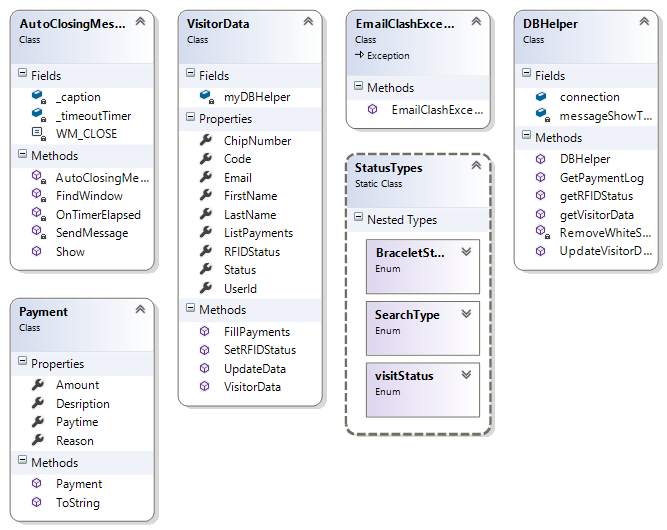
### Visitor support

#### Overview

This application is based on the bracelet management application discussed earlier and is a modified version of the latter. It has less functions than the Bracelet Management, but does implement some of the same methods and also additional functionality used in this application.

This application offers the methods to update the visitor information – email, first name and last name; also it makes it possible to see the overview of all the payments a certain visitor had done (service and store including).

#### Class diagram



* Visitor Info:   
  Same as the one in the Bracelet Management.
* DBhelper:  
  This class has a method to update the visitor data – it is performed on a click of a button ‘Update data’ and writes the newly input data from the textboxes back to the database. However, before the changes are done it checks for the ‘clashing’ email. Below is the code snipped for the email clash checking:  
    
  “...   
   command.CommandText = "SELECT USER\_ID FROM VISITORS WHERE EMAIL = '" + email + "';";

MySqlDataReader reader = command.ExecuteReader();

bool emailClash = false;

while (reader.Read())

{

if (userid != (int)reader["USER\_ID"])

{

emailClash = true;

}  
 }...”  
  
The whole method   
public bool UpdateVisitorData(int userid, string email, string fn, string ln)  
uses transactions to accomplish the task of not writing the incorrect data and preventing other faults.  
  
The payment overview is retrieved by the following query:

string sql = "SELECT DATE as PAYTIME, TYPE as REASON, PAYSUM as AMOUNT, DESCRIPTION" +

" from serpayments WHERE USER\_ID = " + userId +

" UNION " +

"SELECT stp.PURCHASETIME as PAYTIME, s.STORENAME as REASON, stp.TOTALPRICE as SPENT,'Payment in the store' " +

" from storepayment stp JOIN " +

" stores s " +

" on stp.STORE\_ID = s.store\_id " +

" WHERE USER\_ID = " + userId +

" order by PAYTIME ;";  
It uses the union in order to merge the data from two different tables into one.

* File export:   
  This app has a functionality to save the log of payments generated for a visitor in a separate .txt file. This file then can be sent via email on request etc.

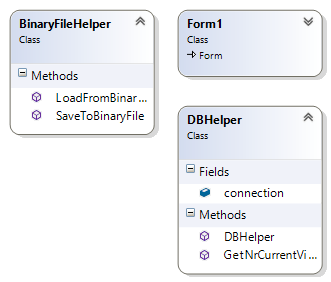
### Statistics app *“StatsApp”*

#### Overview

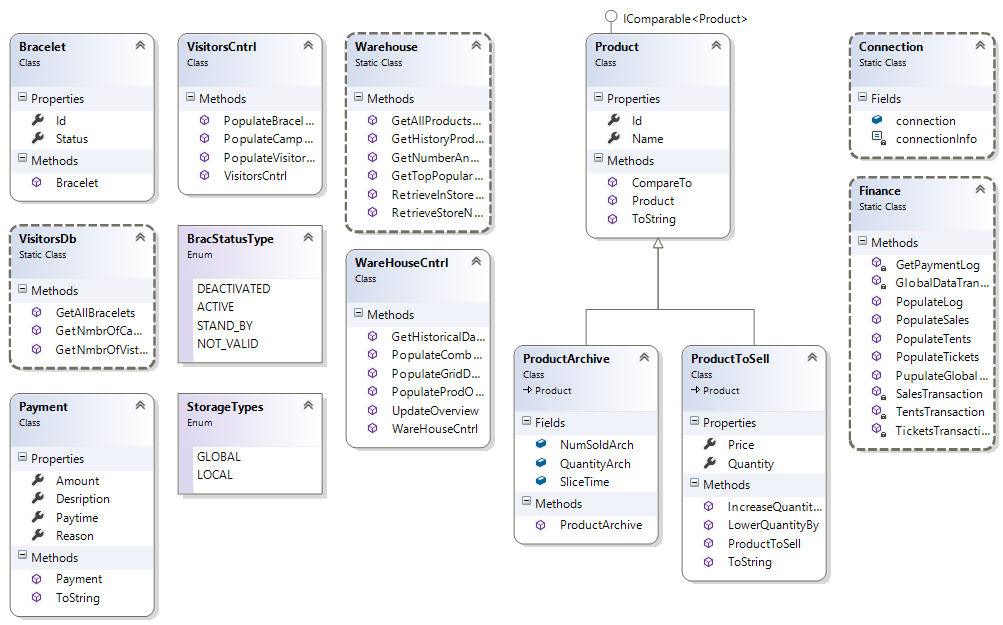
This application is used to keep track of different statistical information that can be retrieved from the database. The different data is managed according to certain patterns. Generally, it has three modules: Visitors, Products and Finances, which present the overviews of the data relevant to the respective modules.

Each module is presented by a separate tab inside of one form. To gain the high level of logic separation inside of the same solution a separate project was created called Modules, which was meant to store the Controllers of the respective modules.   
  
In this application our team had tried to reach the closest to the MVC software architecture concept.

As it is seen below the class diagram of the classes in StatsApp.csproj is relatively small and counts only three class handling files: BinaryFileHelper, Form1, DBHelper. Some of the methods, handling the data, that are present inside of these classes were used at an early stage of the development and thus stayed untouched.

Class diagram for the ‘VIEW’ project.   


In contrast to the class diagram above the class diagram of the classes of Modules.csproj is considerably larger.

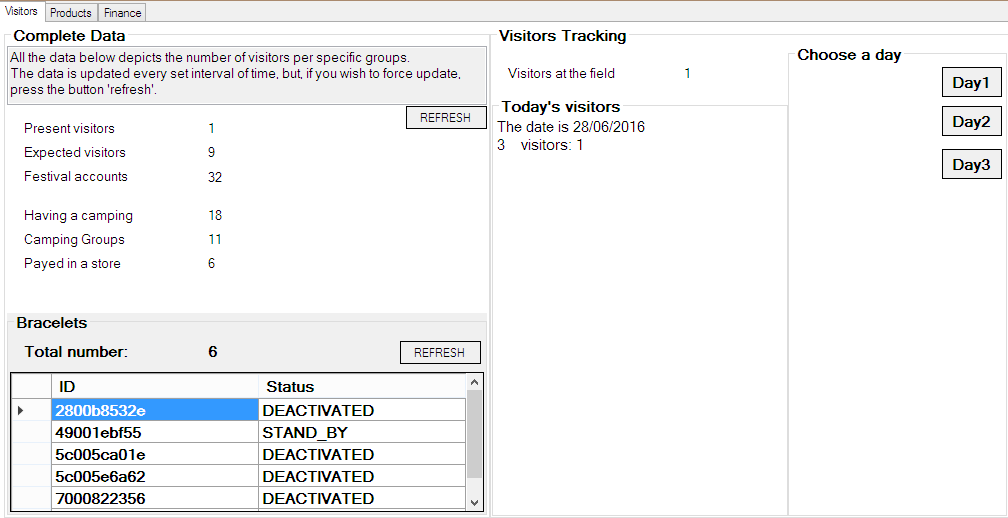


The methods of these classes mostly follow the following assumptions and concepts:

1. Separate all the data manipulation from the form
2. Populate the data to the elements inside of the modules’ methods – thus passing the components as parameters
3. Make the methods to be mostly static

#### Visitors module

This module was the initial module and the first developments were done a long time before the latest updates and even before the whole ‘division of labors’ concept was established. Therefore it partially relies on the methods featured in the StatsApp.csproj and partially – on the final concept. Below is the screenshot of an application at runtime.



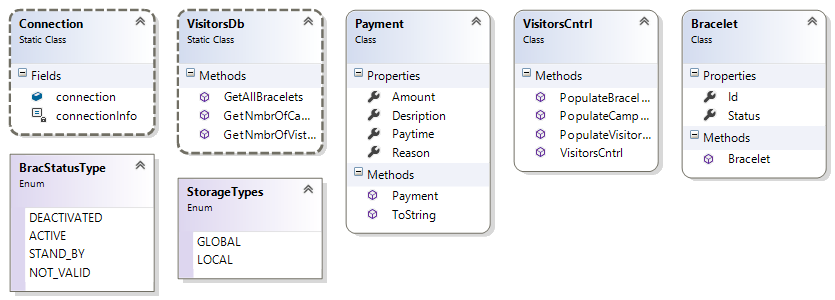
The functionality from the main project involves the two right groupboxes and is based on the class diagram of the StastApp.csproj classes.

The methods rely on the two classes: DBhelper and BinaryFileHelper. The former one provides the methods to retrieve the data from the database: GetNrCurrentVisitors, which returns the number of users who had entered at the event. This method is executed every 100 ms via timer1 of the form1. On every interval of the timer the new data is displayed in the label in the right side of ‘Visitors at the field’ and in the listbox todays’ visitors.

The listbox with 3 buttons on the rightmost side of the window is used to show the data saved for the 1st, 2nd or 3rd day of the event accordingly. On click of a certain day the data from an according binary file is loaded via the BinaryFileHelper’s LoadFromBinaryFile method.

The data is saved as a binary file using the BinaryFileHelper’s SaveToBinaryFile method. The saving is executed every 5 seconds according to the timer2 of the Form 1.

Modules part:



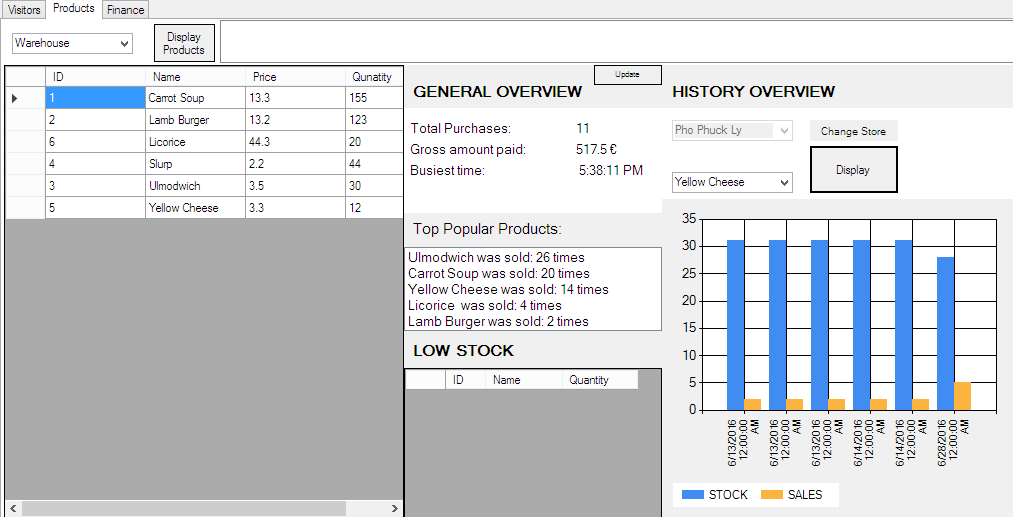
Above is the class diagram of the classes used in the Modules.csproj for the ‘Visitors’ modules. The VisitorsCnrl is used to populate the data by using the methods that take the according components as the parameters and then, by retrieving the data from the database using the methods of VisitorsDB, populates the controls. Below is an example of a pair of methods that populates the numbers for the three following labels. “Present Visitors”(the difference from the current users – is that this number counts only the visitors who have payed and are at the event and the former one counts all – thus it is used to make sure that all the current visitors are the ones who have payed (even though the opposite is very improbable)), “Expected Visitors”, “Festival Accounts”.  
In the left section is the method of VisitorsCntrl and in the right one – of VisitorsDB.

|  |  |
| --- | --- |
| static public void PopulateVisitorGroupData(Label[] lbls)  {  int nmbrTotal, nmbrExp, nmbrPres;  VisitorsDb.GetNmbrOfVistrPerStatus(out nmbrTotal, out nmbrExp, out nmbrPres);  try  {  lbls[0].Text = nmbrPres.ToString();  lbls[1].Text = nmbrExp.ToString();  lbls[2].Text = nmbrTotal.ToString();  }  catch  {  Console.WriteLine("okay, didn't work");  }  } | static public void GetNmbrOfVistrPerStatus(out int nmbrTotal, out int nmbrExp, out int nmbrPres)  {  nmbrTotal = 0;  nmbrExp = 0;  nmbrPres = 0;  ….  DBConnectionDll.Connection.connection.Open();  MySqlDataReader reader = command.ExecuteReader();  while (reader.Read())  {  int.TryParse(reader["TotalVis"].ToString(), out nmbrTotal);  int.TryParse(reader["ExpVis"].ToString(), out nmbrExp);  int.TryParse(reader["PresVis"].ToString(), out nmbrPres);  } |

\*For the further details refer to the source code and the comments

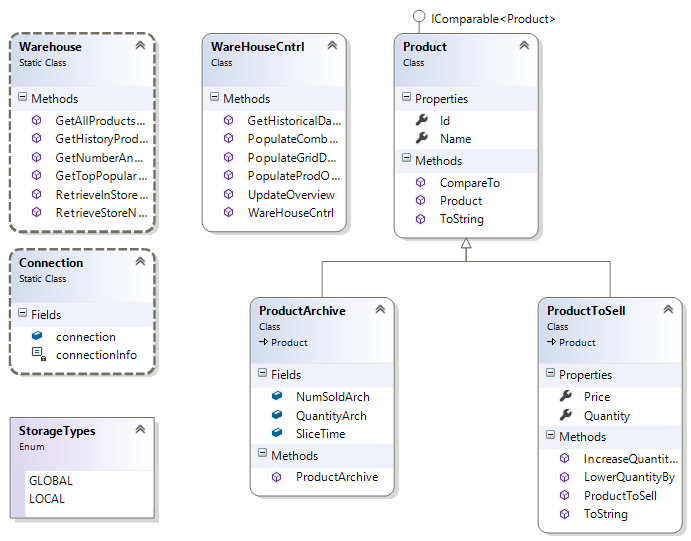
#### Products/Warehouse module

This module is used to keep track of the sales and the product storage conditions – it covers the quantity, number of sales and other statistical data.



The main concept of the separation of tasks between the database interacting and the view populating methods is the same as in the second part of the Visitors Module. This tab makes use of such controls as dataGridView and chart, which makes the tab load a little bit slower.

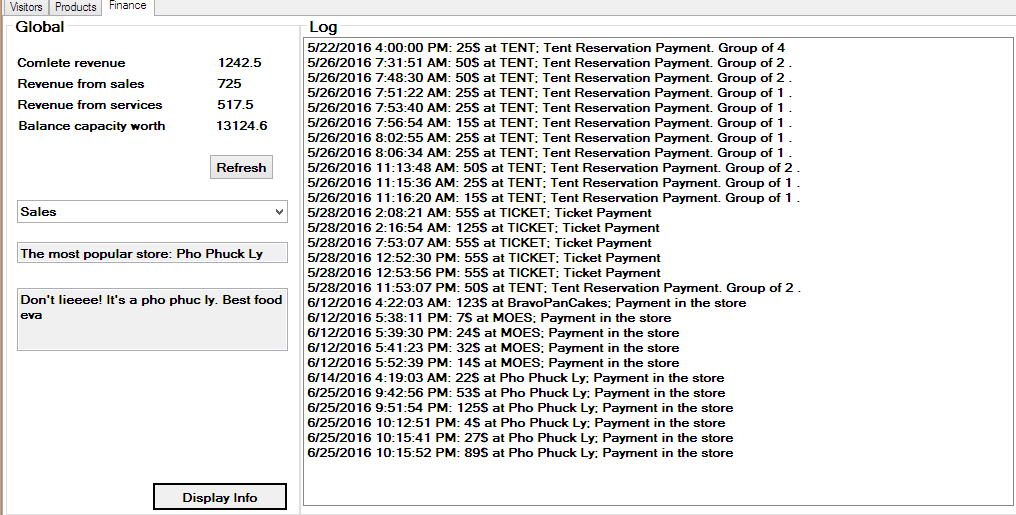
The left part – the largest data grid view is meant for checking the overview of the products distributed per store or stored in the warehouse globally. The latter one retrieves the data of the foodproducts table and the former – of the storeprodinfo view.

The lower grid is populated when the stock of a certain product in the foodproducts table gets too low. 

The chart on the right is built upon the data of the saleslices stored in the storeperfarchive and the data from the current\_quan\_view. For this purpose two types of the product objects are present – two classes: ProductArchive and ProductTotSell are used. They both inherit from the main class Product that inherits from the IComparable interface.

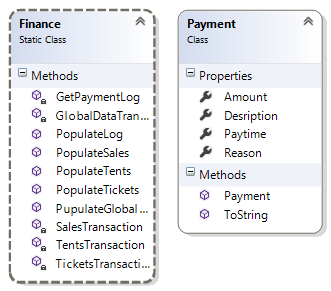
#### Finance module

This module is devoted to keeping the track of the financial data. It presents the overview of the total money earned from the events, shows the data relevant to each type of payment – store and service (divided into both tents and tickets).



The left part is refreshed every time a finance tab is opened or the button refresh is pressed.  
The lower panel featuring the combobox with three options: Sales, Tickets and Tents is used to show the data about the chosen type of payments in the textboxes below. The log on the right is used to present the data about all the payments ever performed at the event (of all kind). The method used is deprived from the Visitor Support application as well as the payment class.

This module is more concise in terms of classes used grace to using the MySQL transactions to retrieve most of the grouped data at once. This facilitates the work with the data retrieved from the database and the way it is retrieved.



Below is the example of the codes that are used to retrieve the data about the tents from the database:

private static List<string> TentsTransaction()

{

List<string> financialTentsValues = new List<string>();

using (MySqlConnection connection = DBConnectionDll.Connection.connection)

{

connection.Open();

MySqlCommand command = connection.CreateCommand();

MySqlTransaction transaction;

// Start a local transaction.

transaction = connection.BeginTransaction();

// Must assign both transaction object and connection

// to Command object for a pending local transaction

command.Connection = connection;

command.Transaction = transaction;

try

{

command.CommandText = @"SELECT SUM(PAYSUM) as tent FROM serpayments WHERE TYPE = 'TENT'; ";

MySqlDataReader reader = command.ExecuteReader();

if (reader.Read())

{

string tentsTotalSum = "Total money sum of tents: " + reader["tent"].ToString();

financialTentsValues.Add(tentsTotalSum);

}

reader.Close();

/////add number of updates

// Attempt to commit the transaction.

transaction.Commit();

Console.WriteLine("Both records are written to database.");

}

catch (Exception ex)

{

Console.WriteLine("Commit Exception Type: {0}", ex.GetType());

Console.WriteLine(" Message: {0}", ex.Message);

// Attempt to roll back the transaction.

try

{

transaction.Rollback();

}

catch (Exception ex2)

{

// This catch block will handle any errors that may have occurred

// on the server that would cause the rollback to fail, such as

// a closed connection.

Console.WriteLine("Rollback Exception Type: {0}", ex2.GetType());

Console.WriteLine(" Message: {0}", ex2.Message);

}

}

}

return financialTentsValues;

}

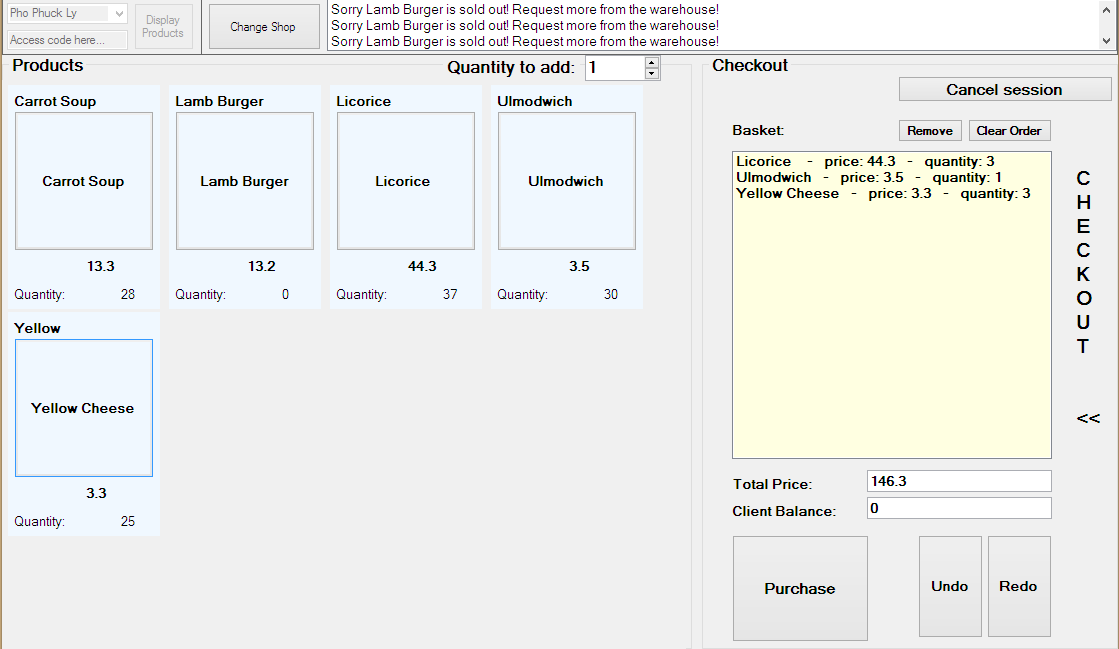
The method above returns the values to be assigned to the labels in one of the populate data methods. Some of the transactions could be replaced by the simple queries as of the time being, but it was done in regards to the continuous development so that more queries could be added.

\*The application is not yet in its potentially perfect state but has some room for easy development in the future.  
\*\*For the additional details and information refer to the source code and comments.

### Shopping application

#### Overview

This application is meant to be used at the shops and the other points of sales. It offers an easy way to sell products to the visitors. No cash is used during the payment – only the visitors’ balance is sufficient and required condition.

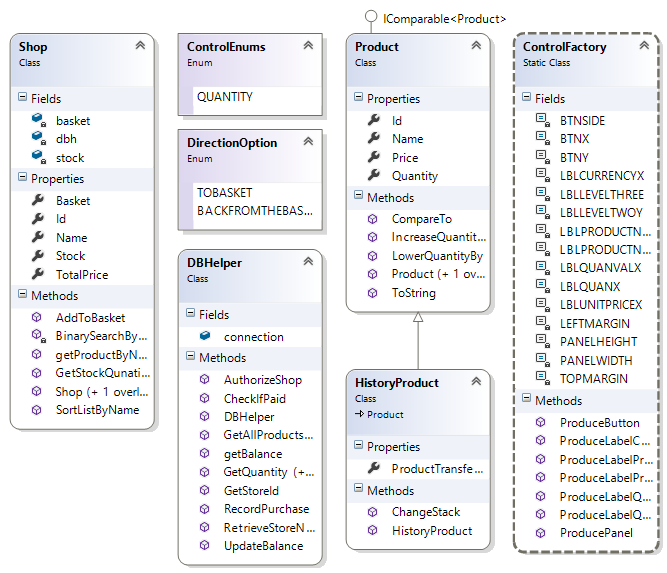


Above is the screenshot of the application at the runtime. As it can be seen the data about the project is retrieved from the database for the according shop chosen in the upper left corner. The upper right corner is used by the log listbox that logs all interactions with the application and the products. Current messages inform that: the “Lamb Burger” is out of stock, and it could not be added to the purchase.

The right panel “Checkout” is used to manage the payment session. The visitor scans the bracelet and his balance is displayed in the current balance text box. A new tag can be scanned only upon pressing “Cancel session” button. Button purchase is clearly used to proceed with a purchase and writes all the changes to the database. Buttons Undo and Redo are used to manage the additions and removals of products to and from the shopping cart. “Clear oder” empties the cart and button “remove” is used to remove a selected item from the basket.

The quantity of the products in the left panel “Products” is managed dynamically and responds to the actions during the session. Until the Purchase button is pressed no changes are written to the database. When it is pressed all the tables that were used in the payment are updated: storepayments, salelines, visitors (balance), productstorages.

#### Class diagram



Above is presented the UML class diagram of the classes used in this application.

* Of a particular interest is the class Control factory that is used for creating the forms of products and populating them inside of the Products panel.  
  The constant properties of this class are used for adjusting the properties of the created controls.   
  Below is an example of code used to create a panel for a product  
    
  public static Panel ProducePanel(ref int X,ref int Y)

{

Panel myPanel = new Panel();

myPanel.Location = new System.Drawing.Point(X, Y);

myPanel.Size = new System.Drawing.Size(PANELWIDTH, PANELHEIGHT);

//myPanel.Name = "pnlTest";

myPanel.BackColor = Color.AliceBlue;

myPanel.Font = new Font(new FontFamily("Microsoft Sans Serif"), 9.75F, FontStyle.Bold);

// moves the starting point of a next panel to the right on an accodring distance

X = X + PANELWIDTH + LEFTMARGIN;

return myPanel;

}

* The difference between a normal product class and a history product is done using the inheritance. The history products are used for redo and undo buttons.   
  For instance when the product was first added and then ‘undone’, when the button redo was pressed it would be restored and added back to the basket – thus the original product has a constructor that creates a copy based on the history product: public Product(HistoryProduct hp)
* In the searching and sorting of products the binary search methods are utilized.

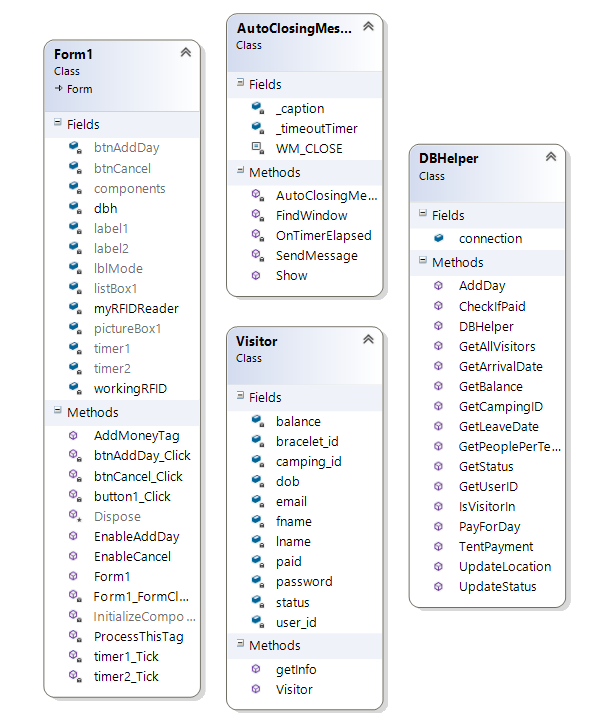
### Camping Entry and Exit

#### Overview

The camping entry and exit apps have similar functionalities. On RFID scan the app gets information about the person with the same RFID number. It does a couple of checks which are described in the app as comments. If everything is in order the location of the person is updated in the database and a color is shown representing a door – green means open, red means do not open.

The camping entry app also has the functionality to extend the reservation of a tent by a day. When a person presses the button for adding a day the scanning executes other checks and methods which are described in the app as comments as well.

#### UML Diagram



#### Notes

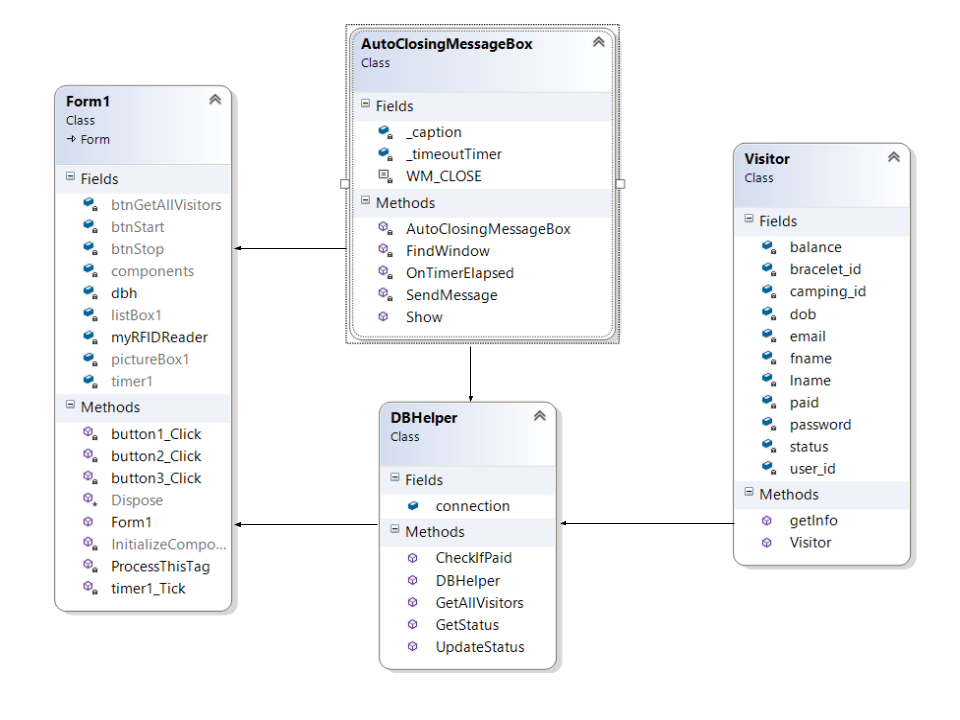
* The camping entry application uses event delegates to switch between entering and reserving an extra day functionality
* The two timers prevent scanning more than one RFID at a time for both functionalities
* The RFID reader should be plugged in beforehand, because the application activates it on form load and deactivates it on form close

### Event Entry and Exit

#### Overview

The entry and exit apps have similar functionalities. On RFID scan the app gets information about the person with the same RFID number. It does a couple of checks if the RFID is registered and if the person has paid. If everything is in order the status of the person is switched between active and inactive and a color is shown representing a door – green means open, red means do not open.

#### UML Diagram

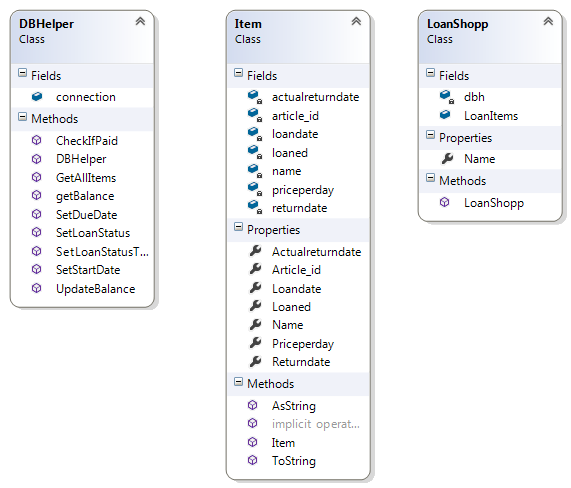


### Loaning Item Application

#### Overview

The loaning items application is used for loaning certain items which the event offers for a certain period of time. When the app is starded you can click on the load button which will load all the available items from the database and display them in a listbox, after that the employee can press the scan button which will activate the RFID Reader and find the person that wants to loan items’s detail via his bracelet string. The next step is to choose an item which he wants to loan and upon clicking the loan button it will interact with the database and set the user\_id to the corresponding one and the loanstatus of the item to 1 (true). After choosing a date from the datetimepicker and clicking the button loan the price will be calculated based on the days which the item will be loaned for times the price per day of the item and it will be substracted from the balance, if the balance of the visitor is insufficient a corresponding messagebox will appear. When the user wants to return an item he can give it back and the employee can choose the item which will be returned based on the article\_id and it when clicking the button return that will use the method SetLoanStatusToFalse from the DBHelper.

#### UML Diagram



#### Extra information

As visible above the DBHelper class has a method (GetAllItems) for retrieving all the items from the database and inserting them in a list of items with their respective properties. The other methods are mostly self-explanatory, the most important methods are the SetLoanStatus() and SetLoanStatusToFalse(), the first of which updates the loanstatus to 1 and sets the user\_id to the corresponding one via the visitors table using the bracelet\_id and the latter one is just nullifying all the values of the user\_id and the loanstatus, when the method is used, which is when returning back an item. The properties of the Item class are all based on the database and are assigned values from it.

public void SetLoanStatusToFalse(int articleid)

{

string sql = "UPDATE loanitems SET LoanStatus = 0, USER\_ID = 0 ,RETURNDATE = NOW() WHERE ARTICLE\_ID = ?Article\_id;";

MySqlCommand command = new MySqlCommand(sql, connection);

command.CommandText = sql;

command.Parameters.AddWithValue("?Article\_id", articleid);

try

{

connection.Open();

MySqlDataReader reader = command.ExecuteReader();

}

catch (Exception exc)

{

MessageBox.Show(exc.ToString());

}

finally

{

connection.Close();

}

}

public void SetLoanStatus(int articleid, string bracelet\_id)

{

string sql = "UPDATE loanitems SET LoanStatus = 1, USER\_ID = (SELECT visitors.USER\_ID FROM visitors WHERE BRACELET\_ID = ?bracelet\_ID LIMIT 1) WHERE ARTICLE\_ID = ?Article\_id;";

MySqlCommand command = new MySqlCommand(sql, connection);

command.CommandText = sql;

command.Parameters.AddWithValue("?Article\_id", articleid);

command.Parameters.AddWithValue("?bracelet\_ID", bracelet\_id);

try

{

connection.Open();

MySqlDataReader reader = command.ExecuteReader();

}

catch (Exception exc)

{

MessageBox.Show(exc.ToString());

}

finally

{

connection.Close();

}

}

### Activity entrance application

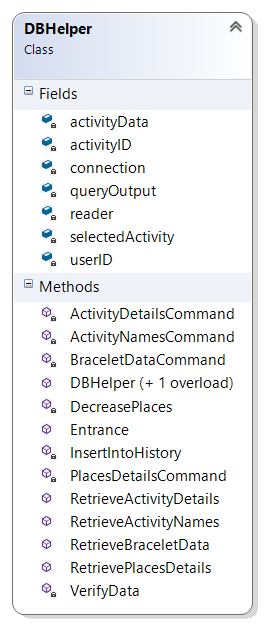
The purpose of this application is keep track of all the visitors that enter a given activity. It features reserved places tracking, free to enter places tracking. It interacts with the database and validates all the data of visitors that are entering current activity and notifies the worker operating the computer.

Finctionality and user expirience:

1. The user starts the app and choosees the activity that is he going to be incharge of at the given moment. What you can see is that data about the selected activity is being populated to the form, which serves as аn informational mean for the user.
2. After he opens the reader and scans some of the chips, he can see that the data about the given bracelet is populated on the form.
3. The next step is to click the proceed button which is going to execute several methods and queries. That is going to result the entrance of the current visitor.

General structure and organisation:

The application consists of two forms connected to each other.



Here can be seen the structure of DBHelper class it is used to interact with the database.

The fields in the class serve as temporary holders of the values used in the method executions

The specific about the structure of the class is that it is divided into two kind of methods:

* Methods that return a mysql command
* Methods that execute these commands and return the retrieved data which is being attached to the visual part of the form afterwards.

The methods that are more specific are the ‘Entrance’ and ‘VerifyData’.

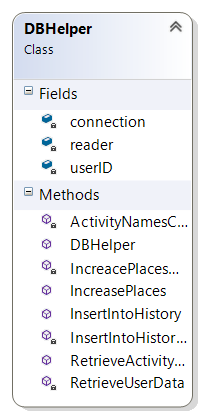
* ‘VerifyData’ acts as a protection against users who are trying to enter illegaly using the history table(it is described in the database specifications)
* ‘Entrance’ executes all the main queries for manipulate the seat numbers.

### Activity exit application

The purpose of this application is keep track of all the visitors that exit a given activity. It interacts with the database and lowers the current number of visitors.

Finctionality and user expirience:

1. The user starts the app and choosees the activity that is he going to be incharge of at the given moment. What you can see is that data about the selected activity is being populated to the form, which serves as аn informational mean for the user.
2. The next for the user is to make sure a bracelet is scanned and click the button ‘proceed’ which is going to execute the relevant queries.



Here can be seen the structure of DBHelper class it is used to interact with the database.

The fields in the class serve as temporary holders of the values used in the method executions

The specific about the structure of the class is that it is divided into two kind of methods:

* Methods that return a mysql command
* Methods that execute these commands and return the retrieved data which is being attached to the visual part of the form afterwards.

# Project Plan:

## Overview:

Over the duration of 19 weeks, our team will develop a new festival control system for Mr. George. This system is meant to provide services both for the attendees and the organizers. It will rely on modern technologies and cover most of the event management aspects. Our team will implement the full-stack development of the system. It will provide an easier access for the pre-festival registration for the visitors, during-festival access management and data tracking.

## Formal client:

Mr. George.

An event administrator from a major event management company based in The Netherlands with a large history of successfully organized festivals. His company is focused mainly on music-related events.

Communication between the Soft-Step team and the client is established via our project coordinator.

For any additional information about the client contact:

Bert Van Gestel.  
 Organisation: Fontys Hogeschool ICT / Engelstalige opleiding  
 Email: b.vangestel@fontys.nl

Location: Rachelsmolen 1 Eindhoven

Telephone number: 08850 71186

## Project Team:

Atanas Naydenov

Team Member

Tel: +31 6 83 13 01 57

e-mail: a.naydenov@student.fontys.nl

Dmitrii Orlov

Team Member

Tel: +31 6 83 13 06 38  
 e-mail: dmitry-orlov@hotmail.com

Dimitar Markov

Team Member

Tel: +31 06 83 14 30 87

e-mail: dmarkov23@gmail.com

Hristian Vasilev

Team Member

Tel: +31 06 58 73 22 31

e-mail: intvasilev1@gmail.com

## Current situation:

Step-Soft is going to develop a full-stack platform for the undersigned client, Mr. George. The latter contacted our group with a request to produce a more efficient access management system for his future events. In our project, the initial version of the system will be developed and used for the music festival “Universe of Sound”. The system will be agile and developed in regard for an easy optimization for the future events organized by the formal client’s company in prospective.

In the system, currently used by the formal client, the ticketing process is organized in the following way.

The advertisement of the event relies on the physical publishing means, i.e newspapers, flyers, leaflets etc. The potential attendees can order a ticket only by telephoning formal client’s company. The only payment methods are the following: a cash payment at the entrance or a direct bank transfer. Once the transfer is processed, the purchased ticket is delivered to the visitor via post. If the visitor opts for the cash payment, the ticket is issued at the cash desk.

All the visitor information is documented manually and is stored in an excel spreadsheet.

No during-the-event access management is implemented. The formal customer’s company cannot track the status of the system and does not have any real time atomized control over the event’s processes like authorization, reservation, entrance, selling etc.

The average number of attendees on the event varies in the range of 4,000 – 50,000 people. The events organized by formal client’s company are mostly music festivals. If the festival is held for longer than one day, a partner company, specializing in camping site management and logistics, usually provides the accommodation on the festivals.

Mr. George intends to replace the current system with our product. He, being a formal client, is at the same time the main project sponsor.

## Project justification:

The client, Mr. George, is currently using a system for organizing his events that he would like to modernize. His company currently relies on traditional ways of marketing, event-management and ticketing. Our formal client would like to introduce modern technologies in organizing his festivals.

Using a more automatized platform relying on modern technologies will not only facilitate the event management process, but also will generate a higher rate of interest in the events, lower the costs, raise the advertisement reach and increase the profit.

## 

## Problem description:

The client wants to try a new event-access management system eliminate the major problems associated to with the currently used one. Thus the system has to be redesigned and replaced by a more efficient, technology-integrated, optimized and self-sustainable platform.

The final product of this project has to cover all the main phases of the event organization and management.

1. Advertisement and social media reach.

2. Registration, ticketing and reservation (accommodation, event activities, etc.)

3. Entrance and event access management

4. During-the-event data control  
 a) Sales

b) Real-time reservations

c) Status tracking

d) System reporting

5. After-the-event data management

## Project goal:

The project’s goal is to develop an electronic festival control system that will be highly user-friendly, accessible, and reusable for the future events, organized by the formal client’s company.

Our team has to analyze the current market and modern technologies to determine the optimal solution for our client.

The final product has to ascertain the web presence of event and establish Internet-based, user friendly technique of purchasing the tickets and make it possible to reserve different facilities provided by the event (such as camping sites).   
 The ticketing method has to increase the entrance rate, minimize the error probability, establish better security and reduce the possibility of any ticket fraud.  
 The event-access management technique has to ascertain the finest attendee experience by providing means of communication, payment, verification of the reservations and status control.  
 The data control services provided for the administration of the event have to be reusable, efficient and help to retrieve and manage the data about the current status of the system, event and finances involved in the event.

## Deliverables and non-deliverables:

Below are stated the main deliverables and non-deliverables for the project. The further, more detailed and additional information can be found in the Project and System Scope and other relevant documents.

### Deliverables:

* Website  
  *Including:*
* event information
* visitor registration
* personal account control
* personal data tracking
* Database
* Covering all the event’s activities and aspects
* Application for controlling the event entrance procedure
* Application for controlling the camping entrance procedure
* Application for the shops and sale points
* Application for the activity reservation
* Application for loaning the necessary equipment for the different event activities
* Application for visitors leaving the event
* Administrative application for data tracking and inspecting the status of the event
* Application for visitor support in cases of identification-related exceptions
* System requirements and documentation
* Presentation about the project and system demonstration
* Setup document
* Process report

### Non-Deliverables:

* Banks integration
* Full stock of necessary hardware
* Facility and general event services (such as medical, security, cleaning etc.) management and status tracking system or software
* Transportation services and logistics
* Stage performance and event program management services or software
* Event accommodation
* Staff activity tracking services or software
* Detailed platform costs, budgeting and financial prospects overview

## Constraints:

Time:

For completing the whole project our team has 19 weeks.

C# programming language:

All the developed software and applications will be built using C# programming language

Operational system:

The final product (Festival Control System) will require the use of Microsoft Windows operated machines.

Accessibility:

The design of the website and the applications has to be highly user-friendly, accessible, and intuitive so that visitors with any level of experience and computer skills could easily operate them.

Reusability:

The design of the system (including all the aspects) has to be self-sustainable and suitable for future implementation in different events without a need of redesigning the architecture.

Documentation:

Each deliverable requires a specification explaining its features, functionality and requirements.

## Risks:

1. Poor time management:  
    An inappropriate scheduling of tasks and activities can make it impossible to accomplish project on time.   
    Probabilty: Moderate  
    Impact on project: Very high  
    Steps to prevent: Use a detailed agenda for tasks and deadlines to guide the team.
2. Poor team dynamics:  
    Low progress in working on the project can lead to a dead end.   
    Probabilty: Low  
    Impact on project: Very high  
    Steps to prevent: Focus on the most important items of the Moscow List first and schedule the work thoughtfully.
3. Poor software quality:  
    Not meeting the quality requirements and not fulfilling the expectations of the formal client.  
    Probability: Moderate  
    Impact on project: High  
    Steps to prevent: Rely on the usual testing and quality checking.
4. Overly optimistic schedule  
    The scheduled tasks can prove themselves to be more time-consuming and complex than expected.  
    Probability: Low  
    Impact on project: High  
    Steps to prevent: Adjust the detailed schedule according to the actual progress, rely on deadlines and prioritize the tasks properly, research the field in search of best solutions and tips.
5. Inadequate designWebsite and software design can differ from the client’s expectations Probabilty: Low.  
    Impact on project: Moderate.  
    Steps to prevent: Base the designs on simplicity, ease-of-use and user-friendliness. Discuss the designs before implementing them.

## Project phasing:

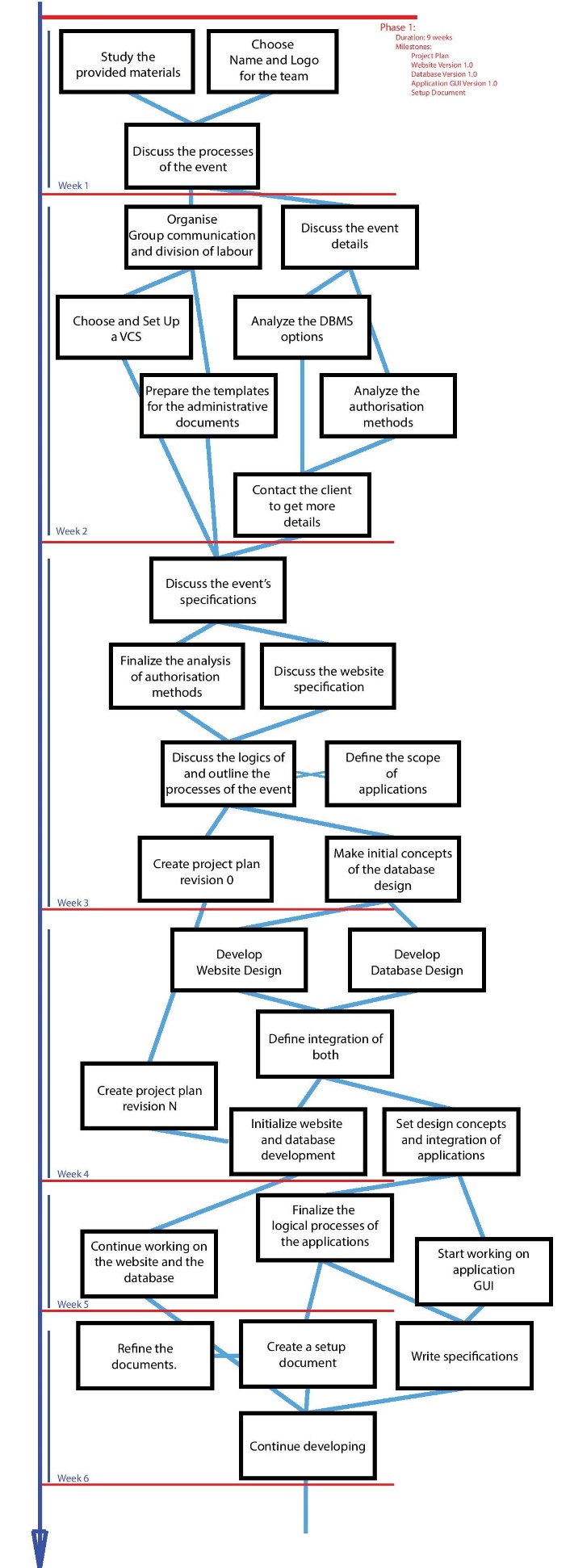
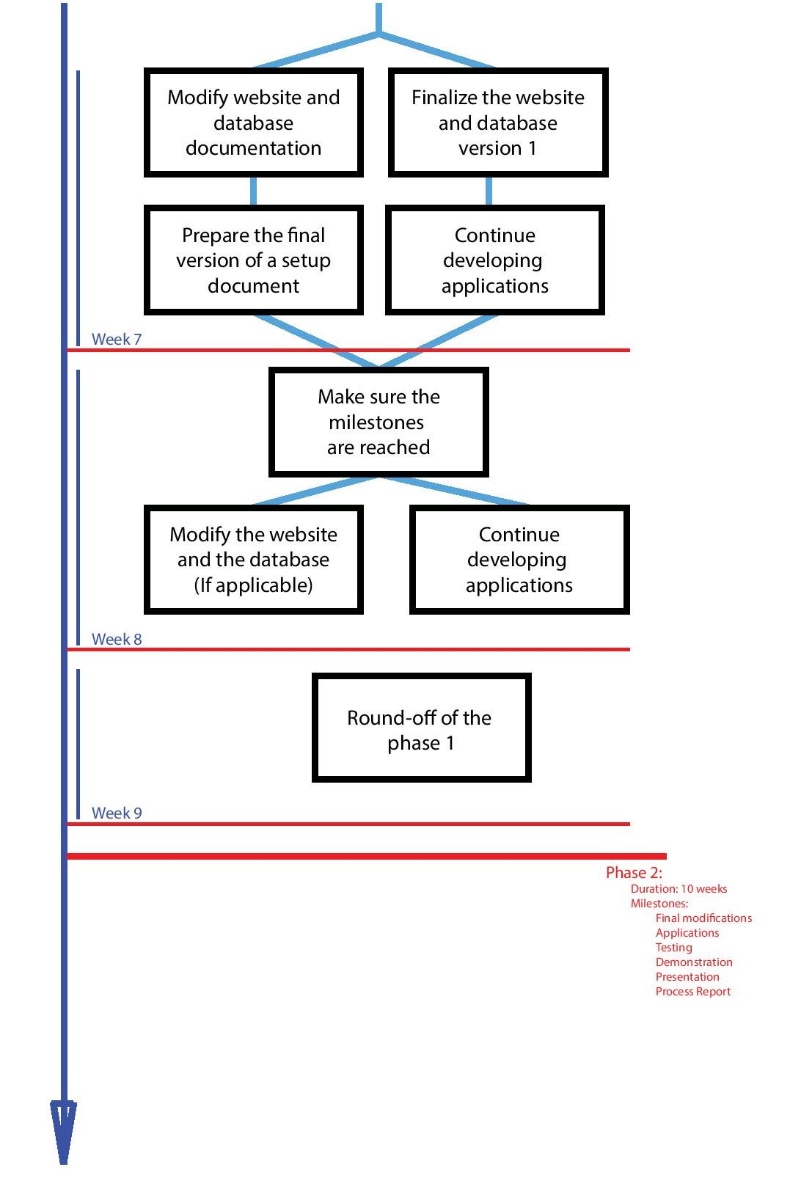
An overview of the phases is present below.  


Figure 1: a) Week 1 – Week 6  
 b) Week 7 – Week 9 + Phase 2

## Phase 1:

Duration: 9 weeks

Milestones:

Project Plan

Website Version 1.0  
 Database Version 1.0

Application GUI Version 1.0  
 Setup Document

Week 1: Initial investigation of the assignment.

Study all the provided materials, define what is unclear and make initial assumptions about the project.

Choose a team name and create a logo.

Discuss the possible functionality and website-database integration patterns and processes involved in the system.

Week 2: Preparations, administrative work and organization

Prepare the templates of the administrative documents and start planning the task schedule, milestone list, communication plan and role schedule.

Discuss the projects uncertainties and reach the client for the discussion.

Introduce the in-group file management techniques and introduce the use of a version control system.

Week 3: Functionality study, option analysis

Discuss the event specifications, prepare the description of the event and included activities.

Analyze the authorization and ticketing techniques.

Refine the processes.

Create a project plan.

Week 4: Initial creation

Determine the website and database design. Define their integration.

Gather media for the webpage and start developing.

Refine the administrative documents.

Define the processes and logics behind the applications and define the integration between applications and the database.

Week 5: Development

Finalize the application processes and logics.

Start working on the application design.

Continue working on the website and the database.

Week 6: Development

Finalize the administrative documents

Continue working on the development.  
 Justify the decisions made.  
 Sketch out visitor-system interaction.

Week 7: Development

Finalize the website and the database.  
 Website and database documentation.

Initialize working on the setup document.   
 Continue developing.

Week 8: <exam week>

Make sure the milestones are reached.

Work on the setup document.  
 Modify the website and database. (If applicable)

On-going development process in all the spheres.

Week 9: <exam week>

Round-off of the phase 1.

## Phase 2:

Duration: 10 weeks

Milestones:

Final modifications (Website, Database)

Applications

Testing

Demonstration

Presentation

Process Report

Week 10: Development, Functionality Study  
 Deploy Website to Athena server

Start implementing functionality of Bracelet Management App and the Event Entry

and Event Exit Apps

Discuss Database design to store additional information for statistics

Discuss Website functionality to populate the Database with additional information

Begin implementation of Personal Page on Website

Week 11: Development

Start implementing functionality of the Stats App (visitors tracking)

Finalize the Event Entry and Event Exit Apps

Test Database and Website interaction

Begin work on Shop App, Item Loaning App  
 Begin implementation of Social Page on Website

Week 12: Development

Continue work on Shop App

Finalize visitor tracking part of Stats App and Bracelet Management App

Continue work on Website

Week 13: Development

Continue work on Shop App, Item Loaning App

Begin work on Camping Entry and Camping Exit App

Continue work on Website

Week 14: Development  
 Continue work on Shop App, Item Loaning App, Camping Entry and

Camping Exit App

Continue work on Website

Week 15: Development

Continue work on Shop App, Item Loaning App, Camping Entry and

Camping Exit App

Continue work on Website

Week 16: Development

Continue work on Shop App, Item Loaning App

Finalize Camping Entry and Camping Exit App

Week 17: Development

Finalize Shop App, Item Loaning App

Finalize Website

Week 18: <exam week>  
 Finalize Project

Test Website and Applications functionality

Week 19: Documentation

Prepare the Final Report

# Work Division Table:

All tasks are divided percentwise, orange is for applications, green is for website, yellow for database and blue for administrative

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Task | Atanas | Dimitar | Dmitry | Hristiyan |
| Camping Entry and Exit Apps | 100% | 0% | 0% | 0% |
| Activity Entry and Exit Apps | 0% | 100% | 0% | 0% |
| Shop App | 50% | 0% | 50% | 0% |
| Item Loaning App | 0% | 0% | 0% | 100% |
| Event Entry and Exit Apps | 100% | 0% | 0% | 0% |
| Statistics App | 20% | 35% | 45% | 0% |
| Bracelet Management App | 0% | 0% | 100% | 0% |
| Website Design | 10% | 10% | 70% | 10% |
| Home page | 30% | 10% | 50% | 10% |
| General Info page | 100% | 0% | 0% | 0% |
| Explore page | 0% | 0% | 100% | 0% |
| Location page | 0% | 0% | 0% | 100% |
| Rules page | 75% | 0% | 25% | 0% |
| Tickets page | 0% | 0% | 65% | 35% |
| Camping reservation page | 0% | 40% | 60% | 0% |
| Activites page | 0% | 0% | 100% | 0% |
| Program page | 0% | 0% | 100% | 0% |
| Registration and Log in | 0% | 100% | 0% | 0% |
| Social page | 0% | 0% | 20% | 80% |
| Personal page | 0% | 100% | 0% | 0% |
| Database Design | 20% | 20% | 40% | 20% |
| Database Modifications | 25% | 25% | 25% | 25% |
| Project Phasing | 10% | 10% | 70% | 10% |
| Project Plan | 10% | 20% | 60% | 10% |
| Setup Document | 20% | 10% | 50% | 20% |
| Final Report | 25% | 25% | 25% | 25% |

# Personal Evaluation:

## Atanas:

I have never worked on a programming group project before, so when we first got into it 19 weeks ago, I must say that I pretty much had no idea what was going on. It took us quite some time to realize how to synchronize (via GitHub) and divide the work between all group members, but eventually all members understood how the whole project was broken down into sub-tasks. Some of the group members already had experience in certain areas so of course a few of their tasks were concentrated in that direction, but that didn’t mean that the others had no say in it. Everybody did a part of everything, so in that line of thought I think we all learned a little bit (or not so little) from each other. There was a number of times where somebody would be working on a task, but land on a stomp or get frustrated with something that he can’t accomplish. In these moments someone else would help him or just pick up from there. In terms of teamwork I think the past few months gave us a lot of experience and knowledge for future projects.   
Time division was also a main concern of the group, because almost every deadline we set had to be extended afterwards. We had a lot of coulds and woulds in the MOSCOW list, that we just didn’t have time to implement, but I guess it was a good thing not implementing the unnecessary things rather than ending up with a core component of the project missing. For me personally, in terms of time management, this project thought me that it is really important to rate all of your ideas by priority and then adjust everything throughout time. This really explains why we do all the documentation for the project.  
When it came to the actual implementation of the project, I can’t even summarize how much I have learned. I can start with the website which was the biggest website I have ever worked on. I refreshed and upgraded my knowledge in html, css and javascript. I also learned a lot of php although I didn’t do much about that part of the website. It was the first time I work on a website that implements a fully functional personal page and has that much of a complex back end part. Regarding the apps first thing that comes to mind is the implementation of the readers as an example of how to work with external hardware. Also all the different types of interaction with the database and all the other features that we have implemented make me feel a lot more comfortable with C# as a programming language.  
But it hasn’t all been smooth sailing. A lot of times we encountered problems like: not having enough knowledge or experience to do what we wanted, not having enough time to do it, pure hardware malfunctions, our work didn’t meet our quality expectations and all sorts of other unexpected errors that tried to stop us on our way to finalizing the project, but all they did was slow us at worst. And I think that’s the most valuable thing we got from this project – How to face and successfully overcome the challenges that a project brings with itself.  
In conclusion I will say that after finishing this project I definitely feel like I have brought my knowledge and capabilities to the next level.

## Dimitar:

I’ll just go ahead briefly with this one, the thing I want to say about this project is that it was extremely useful. I learned a lot about and the most important skill for me - working in team, dividing the task among the different people and keeping track of the workflow. I believe our team went thought a lot of hardness but overall we managed to figure it out. I learned about how to create effective database design and combine and connect desktop and web applications together. I am definitely sure that the gained skills are going to be useful for the future.

## Dmitry:

Over the period of working on this project, I have acquired the experience that I could have obtained in no other way. These 19 weeks have given me such a deep insight over the real project lifecycle and development process. The new knowledge and skills I have received include not only the skills applied purely to programming , but also team management and teamwork in general, time scheduling, design, organization, leadership and has even given an insight into the team psychology.

Especially this project was useful as a mean of getting an insight of a real working environment of a team project. I have never been involved in a project of such large scope before.

It included all the parts of real development – feasibility studies and analysis, design, development, evaluation and constant team management. What was special about this project is that once it’s coming to an end it is so easy (and so frustrating) to see all the fault and misassumptions done before. As the project evolved, the knowledge and skills also developed. A very important part of it was the fact working in a team – it is the biggest source of information and knowledge due to sharing of information and discussing new ideas.

Personally I have acquired the most knowledge about the technical part of the project: i.e. JavaScript frameworks, PHP, code conventions and good practice code, C# techniques and general software architecture concepts from my team-member-friends. Of course, always it was the two way information flow and we learned from each other.

One of the main concerns during the whole process was the way of organizing work, time scheduling, maintaining the team spirit and keeping the goal set – I’ve learned a valuable lesson based on my personal faults and mistakes, most of the things stated further have caused the most difficulties during the project and thus taught the most.

1) Time is scarce.

2) The leadership pattern should be established

3) Design is incredibly important

4) The use of scheduling techniques, task completion analysis and proper evaluation tools is necessary.

5) Enjoy the teamwork.

6) Never assume that something was understood – make sure.

7) Always have a clear goal in mind.

8) Never think that there is too much work – there will always be more.

9) Do it yourself.

10) Have perfectionism in mind (both its benefits and drawbacks).

11) Don’t get distracted.

12) Keep up the good spirit and enthusiasm.

I have greatly enjoyed working in a team and this experience is incredibly valuable. After going through it I understand how important is the friendly atmosphere, common spirit, team support and discussion. I am happy that I was lucky to get a team with which I have enjoyed working. What is the most gratifying I get from this project right now is the evaluation of mistakes and understanding what can be improved and what more should be learned. After this project I have more ideas and a much better understanding of what to avoid further, what to keep and what to improve (mostly improve).

## Hristiyan:

During these 3 months of working on the project I have gained a lot of knowledge. It has been a long and hard trip in which me and my colleagues at Step-Soft have been involved in. The challenge has been enourmous and we have been through thick and thin but we have always tried to help each other with any troubles that we’ve had. I’m very glad to be a part of this group and to have worked on this project and we will continue to expand our enterprise in the future. Most of the time it has been a great experience, of course there were moments of discourage and disappointment, which is inevitable of course but overall it was a very beneficial experience. Even the nights where we stayed up all night were building our character as a team.

Personally, I would improve on the workload which I can take up and probably put even extra time and effort in the next project because sometimes I struggled for longer periods of time with different parts of the project, but nonetheless most of the times my partners have helped me figure out my woes. In addition I have learned a lot about web design, C# programming, especially with the RFID Reader it was a really new experience to me but I’ve enjoyed it and all other parts of working with my team despite all the hard moments.