

Kyle Harrison & Candice Stewart

**2013**

**Teddy Bear Club Nursery Relational Schema**

The Prototype Application for the Teddy Bear Nursery is for administrators and users of Teddy Bear Nursery Application.

**Teddy Bear Club Nursery Relational Schema**

**AC32004–Database Systems**

**Kyle Harrison**

**Candice Stewart**

User Guide – Basic Search and Refine Search

# 1) Basic Search and Refine Search

1. Initial view
2. View all Children in Database
3. Performing a basic search
4. Order by Name
5. Order by Date Of Birth
6. Order by Room Name

# 2) View Search Results

1. Search view
2. View individual child details
3. View Individual Child Consent Details
4. View Individual Child Doctor Details
5. View Individual Child Parents Details
6. View Individual Child Emergency Contacts Details

# 3) Editing and Locking

1. Editing View
2. Unlocking a tab and updating the Database
3. Locking a tab

# 4) Medical Emergency

1. Print Medical Details

# 4) Technical Annex

1. Users
2. Views
3. Stored Procedures

# Basic Search and Refine Search

# Initial View

Upon successful installation of the Application and running the application for the first time the user will be presented with the following user interface.

Search Functionality is included on the left hand side of the page and child information is stored on the right side of the page. Most searching related tasks are automated and very little user input is required.

Buttons adjacent to labels and text boxes have actions that correspond to those attributes.

# View all Children in Database

1. Click on either the First Name or Surname Text Box
2. The list box will update with all children in the database in alphabetic order.

# Performing a basic search

1. Click on either the First Name or Surname Text Box
2. Enter a character to begin a search. E.g. R or r will begin searching for all children with either a name starting with the letter R or any child with an R in their first name

# Ordering all Children by Name

1. If no search has been performed repeat step 1 in “*View All Children in Database*”. If not skip to step 2.
2. You should see the list view updated with all children in the database
3. Select Name from Order By above List View

# Ordering all Children by Date Of Birth

1. If no search has been performed repeat step 1 in “*View All Children in Database*”. If not skip to step 2.
2. You should see the list view updated with all children in the database
3. Select DOB from Order By above List View

# Ordering all Children by Room Name

1. If no search has been performed repeat step 1 in “*View All Children in Database*”. If not skip to step 2.
2. You should see the list view updated with all children in the database
3. Select Room Name from Order By above List View

# View Search Results

# Search View

Whenever a search of any kind is performed the list view will be populated with children from the database.

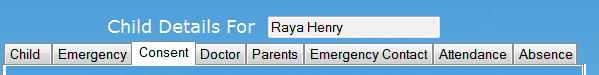
Information on the right side of the page is auto generated whenever a user selects a child from the result in the same interface.

Tab views are used to hide information of less importance but still requires frequent access.

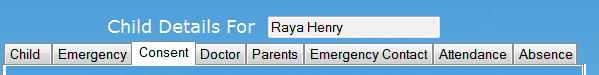
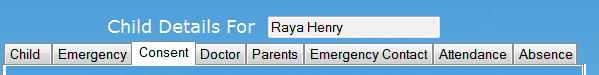
# View Individual Child Details

1. Select a search result from the list view
2. Results will be auto generated for that child on the right side of the screen.

# View Individual Child Consent Details

1. Select a search result from the list view if one is not already selected
2. Select Consent tab
3. Results will auto generate

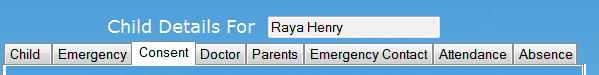
# View Individual Child Doctor Details

1. Select a search result from the list view if one is not already selected
2. Select Doctor tab
3. Results will auto generate

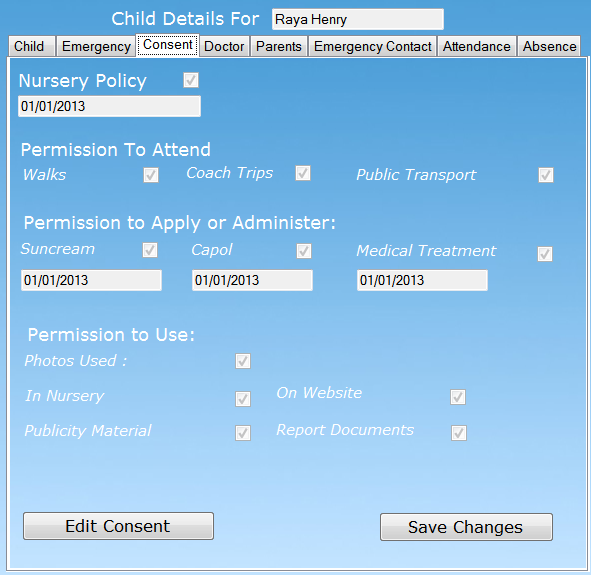
# View Individual Child Parents Details

1. Select a search result from the list view if one is not already selected
2. Select Parents tab
3. Results will auto generate

# View Individual Child Emergency Contacts Details

1. Select a search result from the list view if one is not already selected
2. Select Emergency Contacts tab
3. Results will auto generate

# Editing and Locking

Each tab is locked to editing by default. This means you manually have to enable a button to allow information within the forms to be changed. This is to safeguard against accidental changes to data.

# Unlocking a tab and updating database

1. Once a child has been selected and a tab has been selected
2. Select Edit from the bottom left hand corner to enable editing
3. Make changes to appropriate fields within the form
4. When changes have been made select Save Changes.

Repeat this process for any tab you want to edit.

# Locking a tab

Once a change has been made or a user wants to stop editing

1. If a tab has been unlocked select.
2. Select Lock tab to disable editing.

# Medical Emergency

At anytime there is a medical emergency where information is required quickly. Information can be printed in a minimum of 2 steps and a maximum of 5.

# A) Print Medical Details

1. Click on either the First Name or Surname Text Box. If child is already selected then skip to step 5.
2. Enter a character to begin a search. E.g. R or r will begin searching for all children with either a name starting with the letter R or any child with an R in their first name
3. Select a search result from the list view
4. Results will be auto generated for that child on the right side of the screen.
5. Select Emergency Tab
6. Select Print Emergency Details

# Invoices

For the financing most of the forms are fairly similar. Information is displayed in tables with headings. This however will be changed to a more user friendly view.

By using the tab control it's very easy to manoeuvre through information.









# Technical Annex

# Users

This is an example of the SQL we would use to create users. The passwords would be encrypted for staff and customer security. This would prevent any unauthorised changes being made or data being stolen.

|  |
| --- |
| *USE 12ac3d16;*  *CREATE USER admin [IDENTIFIED BY [PASSWORD] '139003026163048163140058083026136144198161086060']* |

# Creating Views

Creating views was the best way to send information from the database to our application. We only had a few however more will need to be added for full functionality of the application with the database.

|  |
| --- |
| *CREATE*  *ALGORITHM = UNDEFINED*  *DEFINER = `12ac3u16`@`%`*  *SQL SECURITY DEFINER*  *VIEW `individualchilddetails` AS*  *select*  *`child`.`ChildID` AS `ChildID`, `child`.`FirstName` AS `Firstname`, `child`.`Surname` AS `Surname`, `child`.`DOB` AS `DOB`, `child`.`StartDate` AS `StartDate`, `child`.`LeavingDate` AS `LeavingDate`, `child`.`RoomName` AS `RoomName`*  *from*  *`child`*  *group by `child`.`FirstName`*  *order by `child`.`ChildID` desc* |

The above view simply selects certain columns from one table.

|  |
| --- |
| *CREATE*  *ALGORITHM = UNDEFINED*  *DEFINER = `12ac3u16`@`%`*  *SQL SECURITY DEFINER*  *VIEW `monthlyfeedetails1` AS*  *select*  *(`child`.`FirstName` or `child`.`Surname`) AS `Child Name`,*  *`register`.`PlacementType` AS `Placement Type`, `room`.`RoomName` AS `Room Name`, `room`.`DailyCharge` AS `Daily Charge`, `room`.`WeeklyCharge` AS `Weekly Charge`, `attendance`.`Teas` AS `Teas`, `room`.`TeaCharge` AS `Tea Charge`*  *from*  *(((`child`*  *join `register` ON ((`register`.`ChildID` = `child`.`ChildID`)))*  *join `attendance` ON ((`attendance`.`ChildID` = `child`.`ChildID`)))*  *join `room` ON ((`room`.`RoomName` = `child`.`RoomName`)))*  *order by `child`.`Surname` desc* |
| *CREATE*  *ALGORITHM = UNDEFINED*  *DEFINER = `12ac3u16`@`%`*  *SQL SECURITY DEFINER*  *VIEW `monthlyinvoicedetails` AS*  *select*  *(`child`.`FirstName` or `child`.`Surname`) AS `Child Name`,*  *`register`.`PlacementType` AS `Placement Type`, `room`.`RoomName` AS `Room Name`, `room`.`DailyCharge` AS `Daily Charge`, `room`.`WeeklyCharge` AS `Weekly Charge`, `attendance`.`Teas` AS `Teas`, `room`.`TeaCharge` AS `Tea Charge`,*  *`voucherschemeprovider`.`Discount` AS `Discount`, `monthlyinvoice`.`PayDate` AS `Pay Date`, `monthlyinvoice`.`LateFees` AS `Late Fees`*  *from*  *(((((((`child`*  *join `register` ON ((`register`.`ChildID` = `child`.`ChildID`)))*  *join `attendance` ON ((`attendance`.`ChildID` = `child`.`ChildID`)))*  *join `room` ON ((`room`.`RoomName` = `child`.`RoomName`)))*  *join `monthlyfee` ON ((`monthlyfee`.`ChildID` = `child`.`ChildID`)))*  *join `monthlyinvoice` ON ((`monthlyinvoice`.`FeeID` = `monthlyfee`.`FeeID`)))*  *join `voucherschemeregister` ON ((`voucherschemeregister`.`ChildID` = `child`.`ChildID`)))*  *join `voucherschemeprovider` ON ((`voucherschemeprovider`.`VoucherSchemeProviderId` = `voucherschemeregister`.`VoucherSchemeProviderId`)))*  *order by `child`.`Surname` desc* |
| *CREATE DEFINER=`root`@`localhost` PROCEDURE `updateDoctor\_pro`(*  *IN \_DoctorID varchar(6),*  *IN \_FirstName varchar(30),*  *IN \_Surname varchar(30),*  *In \_AddressID varchar(6)*  *)*  *BEGIN*  *UPDATE doctor SET*  *FirstName = \_FirstName,*  *Surname = \_Surname,*  *AddressID = \_AddressID*  *WHERE DoctorID = \_DoctorID;*  *END* |
| *CREATE*  *ALGORITHM = UNDEFINED*  *DEFINER = `12ac3u16`@`%`*  *SQL SECURITY DEFINER*  *VIEW `overduedetails1` AS*  *select*  *(`child`.`FirstName` or `child`.`Surname`) AS `Child Name`,*  *`paymenthistory`.`DateDue` AS `Date Due`,*  *`monthlyinvoice`.`DaysOverdue` AS `Days Overdue`,*  *`monthlyinvoice`.`LateFees` AS `Late Fees`,*  *`monthlyinvoice`.`Overdue` AS `Total Overdue`*  *from*  *(((`child`*  *join `monthlyfee` ON ((`monthlyfee`.`ChildID` = `child`.`ChildID`)))*  *join `monthlyinvoice` ON ((`monthlyinvoice`.`FeeID` = `monthlyfee`.`FeeID`)))*  *join `paymenthistory` ON ((`monthlyinvoice`.`InvoiceID` = `paymenthistory`.`InvoiceID`)))*  *order by `child`.`Surname` desc* |
| *CREATE*  *ALGORITHM = UNDEFINED*  *DEFINER = `12ac3u16`@`%`*  *SQL SECURITY DEFINER*  *VIEW `payhistorydetails` AS*  *select*  *`paymenthistory`.`Payee` AS `Payee`,*  *`paymenthistory`.`Paid` AS `Paid`,*  *`paymenthistory`.`PaymentType` AS `Payment Type`,*  *`paymenthistory`.`DateDue` AS `Date Due`,*  *`paymenthistory`.`DatePaid` AS `Date Paid`,*  *`paymenthistory`.`AmountDue` AS `Amount Due`*  *from*  *`paymenthistory`*  *order by `paymenthistory`.`Payee` desc* |
| *CREATE*  *ALGORITHM = UNDEFINED*  *DEFINER = `12ac3u16`@`%`*  *SQL SECURITY DEFINER*  *VIEW `roomchangedetails` AS*  *select*  *(`child`.`FirstName` or `child`.`Surname`) AS `Child Name`,*  *`child`.`RoomName` AS `Room Name`,*  *`room`.`RoomVacancy` AS `Room Vacancy`*  *from*  *(`child`*  *join `room` ON ((`room`.`RoomName` = `child`.`RoomName`)))*  *order by `child`.`Surname` desc* |

Amongst these views above a majority has had to have INNER JOINS to pull data from more than one table and combine them as a new table. All of the information they display however, is from their original tables.

# Stored Procedures

Due to the application not being fully complete, we haven't got as many procedures as we would actually need, below are the ones we have currently. These procedures access the tables related to Child and allow searches and updates from the application that can access the database and display these results or edit data.

|  |
| --- |
| *CREATE PROCEDURE `proc\_AddChild` (IN FirstName VARCHAR(30), IN Surname VARCHAR(30),*  *IN DOB DATE,IN StartDate DATE, IN LeavingDate DATE, IN RoomName VARCHAR(6), IN AddressID (6))*  *BEGIN*    *DECLARE HighestChildID VARCHAR(6);*  *SET HighestChildID = SELECT MAX(ChildID) FROM Child;*  *SUBSTRING(str FROM pos)*    *IF variable1 = 0 THEN*  *SELECT variable1;*  *END IF;*    *IF param1 = 0 THEN*  *SELECT 'Parameter value = 0';*  *ELSE*  *SELECT 'Parameter value <> 0';*  *END IF;*  *END //* |
| *CREATE PROCEDURE `proc\_FindChildID` (OUT CID VARCHAR(6))*  *BEGIN*    *SELECT MAX(ChildID) INTO @CID FROM Child;*  *SET @CID = SUBSTRING(@CID FROM 1);*  *SELECT @CID;*  *END //* |
| *CREATE DEFINER=`root`@`localhost` PROCEDURE `updateAddress\_pro`(*  *IN \_AddressID varchar(6), IN \_AddressLine varchar(50),*  *IN \_AddressLineTwo varchar(50),*  *In \_City varchar(35),*  *In \_County varchar(30),*  *In \_Postcode varchar(8),*  *In \_Country varchar(30)*  *)*  *BEGIN*  *UPDATE address SET*  *AddressID = \_AddressID,*  *AddressLine= \_AddressLine,*  *AddressLineTwo = \_AddressLineTwo,*  *City = \_City,*  *County= \_County,*  *Postcode= \_Postcode,*  *Country = \_Country*  *WHERE AddressID = \_AddressID;*  *END* |
| *CREATE PROCEDURE `tbnc`.`updateChild\_pro` (*  *IN \_ChildID varchar(6),*  *IN \_FirstName varchar(30),*  *IN \_Surname varchar(30),*  *IN \_DOB DATE,*  *IN \_StartDate DATE,*  *IN \_LeavingDate DATE,*  *IN \_RoomName SET('Baby Room','Polar Room','Panda Room','Teddys Transformers','Daisys Discoverers','Oakleys Explorers'),*  *IN \_AddressID varchar(6)*  *)*  *BEGIN*  *UPDATE child SET*  *FirstName = \_FirstName,*  *Surname = \_Surname,*  *DOB = \_DOB,*  *StartDate = \_StartDate,*  *LeavingDate = \_LeavingDate,*  *RoomName = \_RoomName,*  *AddressID = \_AddressID*  *WHERE ChildID = \_ChildID;*  *END //* |
| *CREATE PROCEDURE `tbnc`.`updateConsent\_pro` (*  *IN \_ConsentID varchar(6),IN \_DoctorID varchar(6),IN \_NurseryPolicies BIT(1),*  *IN \_NurseryPoliciesDate DATE, \_AdministerCapol BIT(1),IN \_AdministerCapolDate DATE,IN \_ApplySunCream BIT(1),IN \_ApplySunCreamDate DATE,*  *IN \_CoachTrips BIT(1),IN \_NurseryPhotos BIT(1),IN \_PhotosTaken BIT(1),*  *IN \_PublicTransport BIT(1),IN \_PublicityPhoto BIT(1),IN \_ReceiveMedicalTreatmentDate DATE,IN \_RecieveMedicalTreatment BIT(1),*  *IN \_ReportPhoto BIT(1),IN \_Walks BIT(1), IN \_WebsitePhotos BIT(1)*  *)*  *BEGIN*  *UPDATE consent SET*  *ConsentID = \_ConsentID ,DoctorID = \_DoctorID,NurseryPolicies = \_NurseryPolicies,NurseryPoliciesDate = \_NurseryPoliciesDate,AdministerCapol = AdministerCapol, AdministerCapolDate = \_AdministerCapolDate , ApplySunCream = \_ApplySunCream,ApplySunCreamDate = \_ApplySunCreamDate, CoachTrips = \_CoachTrips,NurseryPhotos = \_NurseryPhotos,*  *PhotosTaken = \_PhotosTaken, PublicTransport = \_PublicTransport,*  *PublicityPhoto = \_PublicityPhoto,ReceiveMedicalTreatmentDate = \_ReceiveMedicalTreatmentDate,RecieveMedicalTreatment = \_RecieveMedicalTreatment, ReportPhoto = \_ReportPhoto, Walks = \_Walks, WebsitePhotos = \_WebsitePhotos*  *WHERE ConsentID = \_ConsentID;*  *END //* |