Software Design Document for Anahid-Gardens: A nursery management system

Mahmoud Ahmed, John Mounir, Youssef Alaa, David Adel

May 13, 2018

1 Introduction

1.1 Purpose

This software design document describes the architecture and system design of a nursery management system. This document contains the software Architecture , the design choices and the traceability matrix of the system.

1.2 Scope

This document contains the complete description for our system design which is important for experts and developers to understand our system architecture. This system aims to transfer all the paper work to be computerized, it will include an attendance checklist for the children and workers, as well as adding courses, classes, liabilities and creating schedule for each class. It will also manage the payments of subscription, trips and events of the nursery. As well as creating dynamic applications using EAV Model for the applying users(Teachers, kids, workers, etc..). When a new child application is created it will send a notification to the admin. Also the admin can create schedules for the classes, adding courses and lessons to the corresponding courses, and controlling the resources and liabilities of the institute and get a detailed report of every operation done on the system.

1.3 Overview

This is the overview of Software Design Document. the next section provides the system overview and how the system works. Then the system Architecture section have Architectural Design,Sequence Diagram,Decomposition Description (Class Diagram) and Design Rational. After that the Data Design section which have Data Description and Data Dictionary.Then screen images from the system.Finally the last section provides the Requirements Matrix.

1.4 Definitions and Acronyms

This section provides definitions of all terms, acronyms, and abbreviations that might exist to properly interpret the SDD.

Term	**Definition**
Software Design Document (SDD)	Used as the primary medium for communicating software design information.
MVC	Model view controller.
EAV	Entity Attribute Value Model.

2 System Overview

The system is able to manage all the tasks done by the workers in the nursery, such as taking children attendance, checking their activities, making child application, controlling institute resources

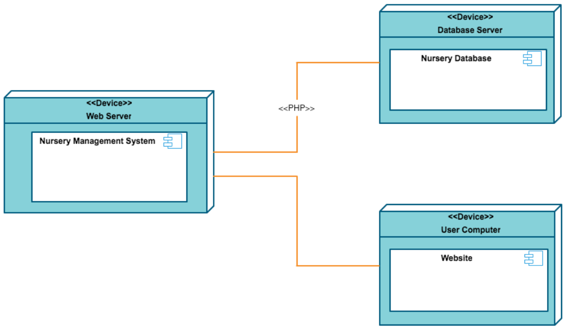


Figure 1: System Overview

3 System Architecture

3.1 Architectural Design

The nursery management system is designed in an MV architecture.

3.1.1 View

This represents the user interface which is divided into three interfaces.

- Admin interface: The admin will login and will have all the views of functionalities available as shown in the architecture diagram.

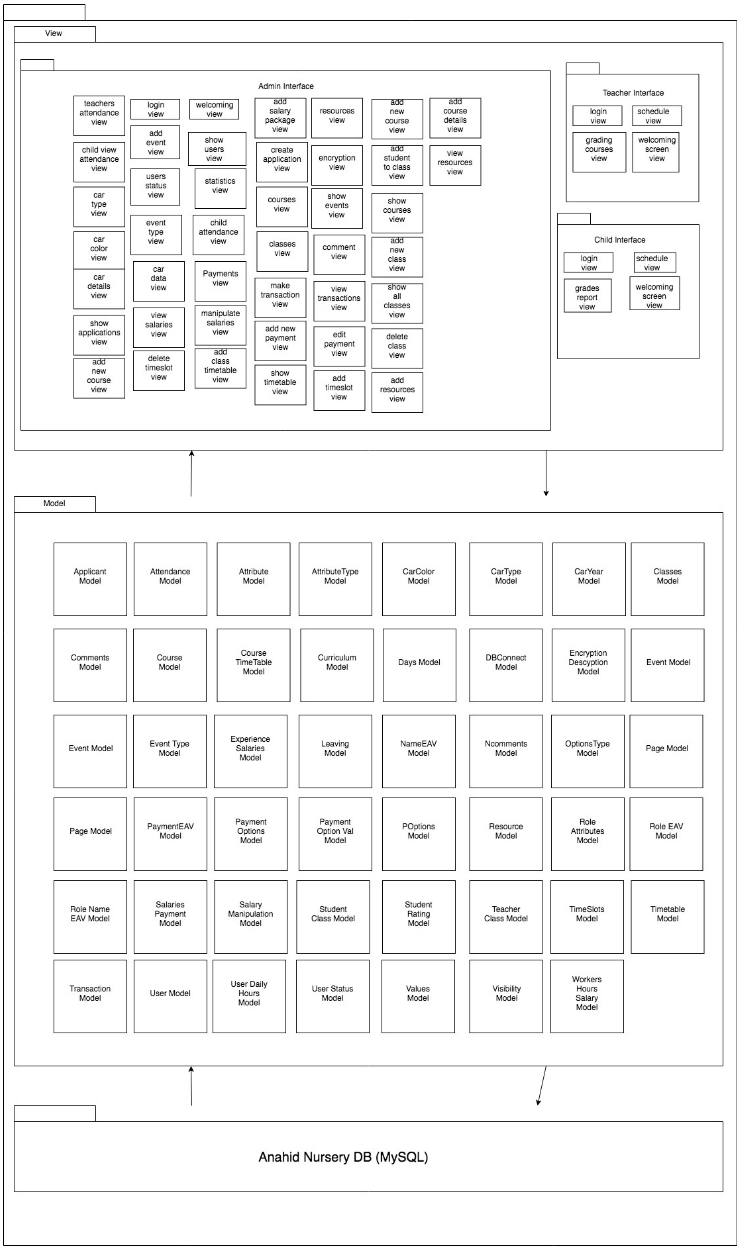


Figure 2: Architectural Design

- Child interface: The child will login and will have a welcoming screen, then a menu showing the schedule and his grades report.

- Teacher interface: Teacher will login and will have a welcoming screen, then a menu showing their schedule showing their assigned class and a screen for grading the students.

3.1.2 Model

Core: The core tables are implemented based on EAV model as shown in figure 2,

3.2 Decomposition Description

3.2.1 Sequence Diagrams

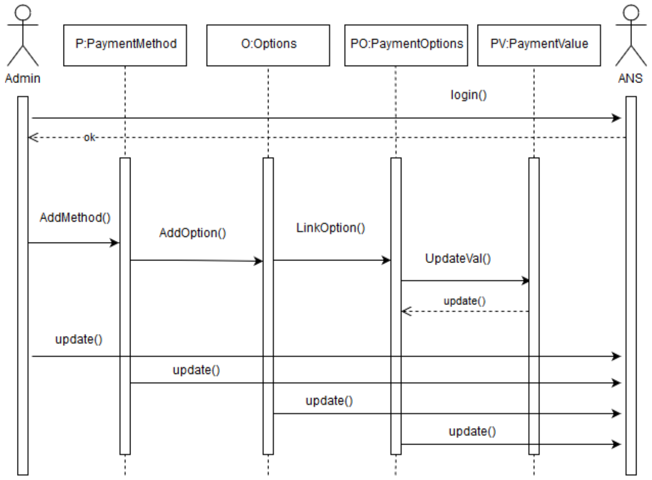


Figure 3: Payments Sequence Diagram

3.3 Design Rationale

The nursery system is based on MV, its architecture diagram is in the MV form. MV is the separation of Model and View classes. Main advantage of MV architecture is differentiating the layers of a project in Model and View for the Re-usability of code and better maintenance.

4 Data Design

4.1 Data Description

This is a relational preview of the database.

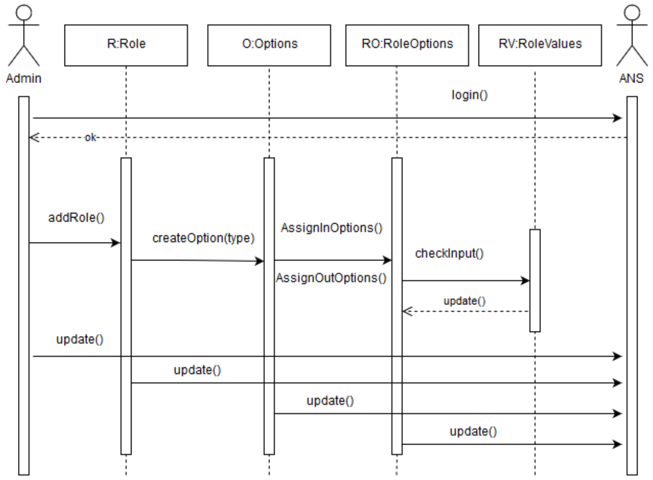


Figure 4: Role Sequence Diagram

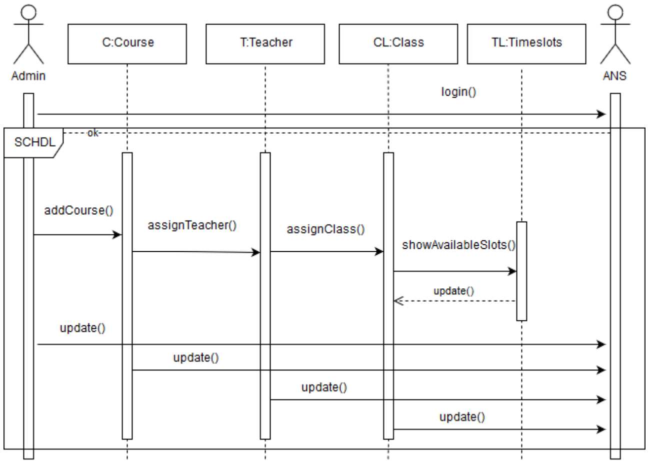


Figure 5: Schedule Sequence Diagram

4.2 Data Dictionary

Alphabeticaly list the system entities or major data along with their types and descriptions. If you provided a functional description in Section 3.2, list al the functions and function parameters. If you provided an OO description, list the objects and its attributes, methods and method parameters.

- **activities**: This is a look-up table carrying the activities done by the childs in the nursery.
- **application**: This is a EAV table carrying the Role ID of the user, Ap-

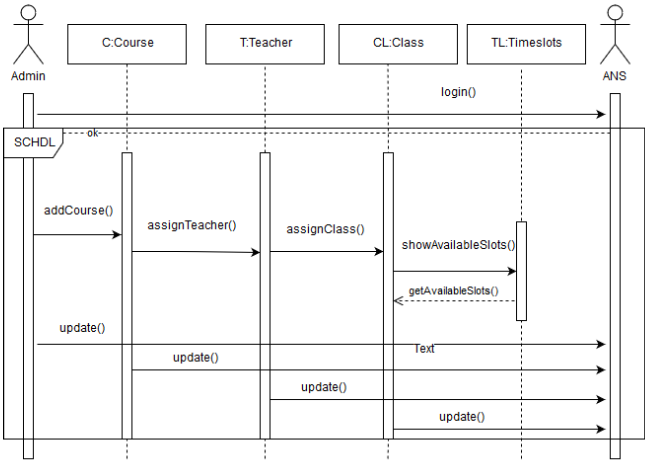


Figure 6: Timeslots Sequence Diagram

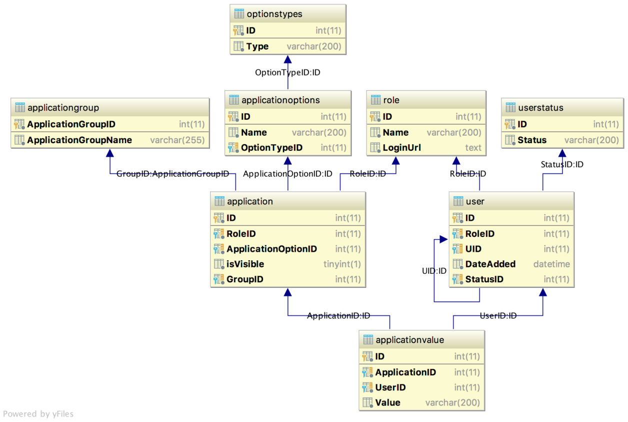


Figure 7: Relational View of the user application EAV model

plicationOptionID which is a foreign key from Application Option, which carries the fields of the application

- **attendance**: The attendance of the childs and teachers is saved with their UserID and datetime stamp, with their status whether attended or not.
- **attendancelog**: The same collection of data from the attendance table, but every month all the records from the first table are moved to the second table to maintain performance.
- **busschedule**: This collection has the UserID which is the driver of the bus, Time : arriving time of the bus to the child, ChildID: the child

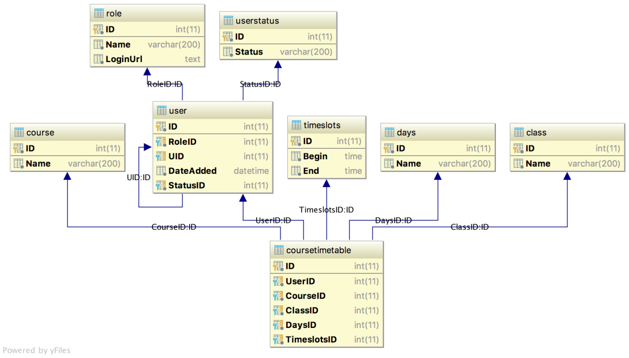


Figure 8: Relational View of the course time table

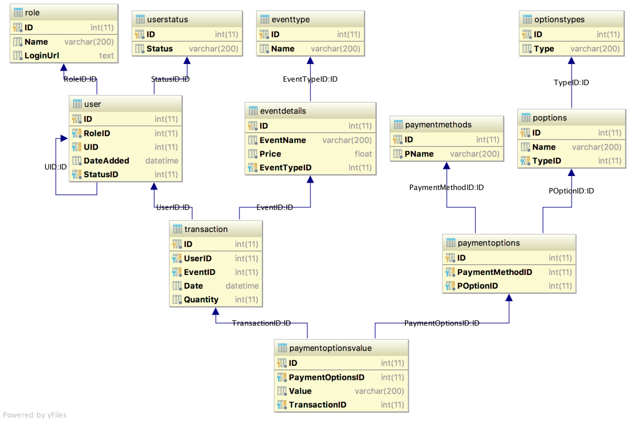


Figure 9: Relational View of the payments

subscribed to the bus, isArriving: if 0, it means the driver should pickup the child from the nursery to home, if 1, it means the driver should pickup the child from home to the nursery at the desired time.

- **carcolor**: This collection is a look-up table of the colors of the cars that the admin can add.

- **cardata**: This collection contains the TypeID: car type and model, ColorID: the color of the car, YearID: the year model of the car, DriverID: the driver which is assigned to this car, PlateNb: the plate number of the car.

- **cartype**: This collection is a look-up table of the car Models and types, containing a self reference ID(CarTypeID) references the model to the car type.

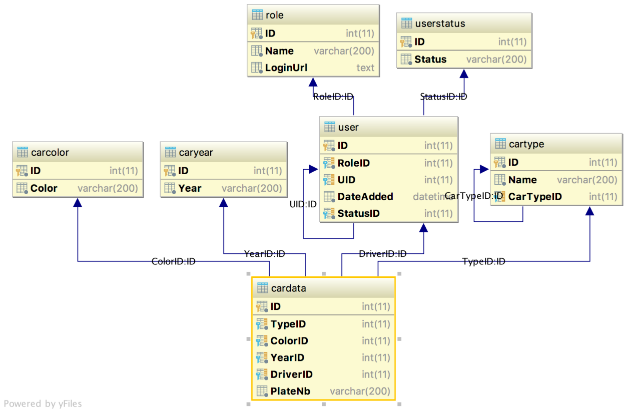


Figure 10: Relational View of the car data

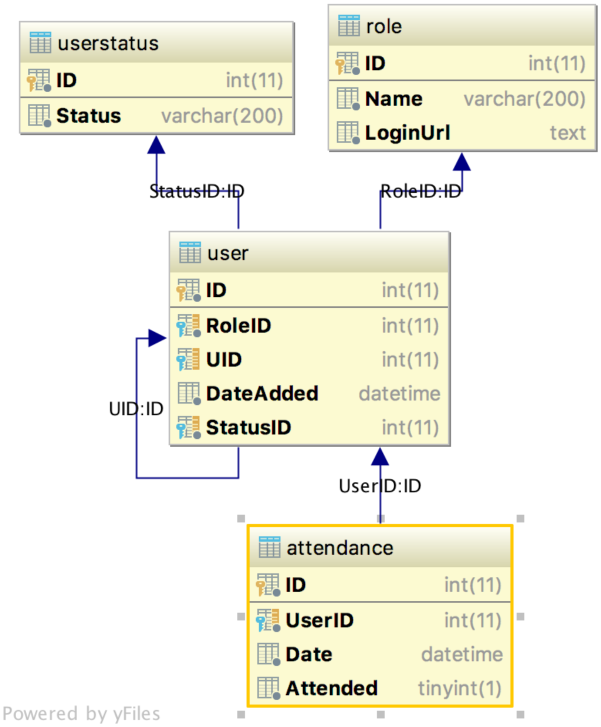


Figure 11: Relational View of the attendance

- **caryear**: This collection is a look-up table of the years the car has been manufactured.

- **class**: This collection is a look-up table of the classes in the nursery.
- **commentdetails**: This collection is a look-up table for the comments done on any application or event , it is separated to prevent nulls in the database .
- **comments**: This collection contains EventDetailsID: the event which has been commented on, Value: the comment itself.
- **course**: This collection is a look-up table of the courses that are being taught in the nursery.
- **coursetimetable**: This collection contains the UserID: the user belonging to this schedule, CourseID: the course id, classID: the class he has been registered to, DaysID: the day of the week, TimeSlotsID: the time slot this section is assigned to.
- **curriculum**: This collection contains CourseID: the course name, LessonName: the lesson assigned to this course, LessonDetails: the description of the this lesson.
- **days**: This collection is a look-up table of the days of the week.
- **disease**: This collection is a look-up table of the diseases the childs may be affected to, so that the medicine may be added to it.
- **duration**: This collection contains the duration the fired, or left Users from the nursery like childs or workers, UserID: which user left, StartDate: the date he/she started with the nursery, LeftDate: the date he/she left, LogDate: the date the admin made changes to this record.
- **errortypes**: This collection is a look-up table of the error types the system might have.
- **eventdetails**: This collection contains the event Name, Price, EventTypeID: the type of the event whether it is a trip, BD, subscription, etc..
- **eventtype**: This collection is a look-up table of the event types.
- **expenditure**: This collection contains the expenditures of the nursery, Value: the price that has been paid, Date: date of expenditure, PaymentMethodID: the method that has been used to make this payment, ExpenditureTypeID: the type of expenditure.
- **expendituretypes**: This collection is a look-up table of the types of expenditure.
- **experiencesalaries**: This collection contains the bonus added to the salary of the user when he's experienced, UserID: the user which will have the increase, Value: the value of the bonus added.

- **extraclothes**: This collection is a look-up table of clothes the child brings with him to the nursery.
- **food**: This collection is a look-up table of the food that is being served to the child Name: name of the food, FoodID: a self reference ID to the parent food.
- **foodies**: This collection keeps record of the food eaten by the child everyday, UserID: the child id, FoodID: the food he ate, Date: date of the day.
- **foodieslog**: The same collection of data from the foodies table, but every month all the records from the first table are moved to the second table to maintain performance.
- **foodtimetable**: This collection keeps the timetable of the food, StartTime: start time of the meal, EndTime: the end time of the meal, FoodID: the food eaten.
- **form**: This collection is EAV model, FormNameID: name of the form, FormOptionsID: the options that will be contained in the form.
- **formname**: This collection is a lookup table keeping the form names.
- **formoptions**: This collection is a lookup table keeping the options of the form.
- **formoptionsvalue**: This collection keeps the value of the options of the form, UserID: the user created this form, FormID: the form containing those options, Value: the value of the option, like first name, last name, username, etc...
- **holiday**: This collection keeps the record of the holiday requested by the workers, UserID: the user requesting the holiday, StartDate: Starting date of the holiday, EndDate: ending date of the holiday
- **leaving**: This collection keeps the leaving time of the teachers for the attendance for the calculation of salaries, attendanceID: the id of the record of the attendance, LeaveTime: the leave time of the worker.
- **localization**: This collection is a lookup table keeping the languages of the system.
- **logtable**: This collection keeps the log of the operations done on the system by the users, Message: containing the operation done by the user.
- **medicine**: This collection is a lookup table of the medicines.
- **medicinedisease**: This collection keeps the medicine issued to every child, MedicineID: the medicine he/she will take, UserID: the child which will take the medicine, DiseaseID: the disease he's affected by, isSick: is he/she still sick or no.

- **messagesanderrors**: This collection keeps the type of the error assigned to each page, Name: name of the message, TypeID: type of the error, PageID: the page corresponding to this error.
- **ncomments**: This collection keeps the notifications of the system, comment-text: the comment of the notification, comment-status: the status of the comment whether it is read or no.
- **nulls**: This collection keeps the nulls to prevent nulls in the other tables, Name: name of the null.
- **optionstypes**: This collection is a lookup of the form options.
- **pages**: This collection keeps the pages of the website, Name: name of the page, URL: the relative URL of the system, ParentID: the parent page.
- **pageshtml**: This collection keeps the html code of the page in the database, PagesID: the page id consisting of this HTML, HTML: the code.
- **paymentmethods**: This collection is a lookup table of the payment EAV model, PName: the payment name: visa, cash, etc..
- **paymentoptions**: This collection keeps the options available of the payment type, PaymentMethodID: the payment method type, POptionsID: the option type
- **paymentoptionsvalue**: This collection keeps the values of the payments, PaymentOptionsID: the payment methodID, Value, the money paid, TransactionID: the transaction assigned to this payment.
- **poptions**: This collection keeps the options the payment method, Name: name of the field, TypeID: the type of the field.
- **report**: This collection keeps the records of
- **reportname**: This collection is a lookup table of the report names.
- **reportoptionstype**: This collection keeps the option type used in the report, OptionsTypeID: the option ID, Name: name of the option.
- **reportsvalue**: This collection keeps the values of the report, UserID: user assigned to this record, ReportID: the report id, Value: the value of the report.
- **resources**: This collection keeps the liabilities of the nursery, Name: name of the resource, Quantity: quantity of each item and quantity of all category, ParentID: the parent category of this resource.
- **role**: This collection keeps the roles of the users in the system with their loginURL, Name: name of the role, LoginURL: the url which will be redirected to on sign in.

- **rolepages**: This collection keeps the pages assigned to each role, RoleID: the role name reference, PageID: the page name reference.

- **salariespayment**: This collection keeps salaries of the workers, ValueToBePaid: the salary itself, isPaid: a boolean check for the status of the payment, StartDate: the start date of the salary, EndDate: the end date of the salary, UserID: the user which will take this salary.

- **salarymanipulation**: This collection keeps the manipulation of the salary, UserID: the worker which will take the salary, Date: date of taking the salary, Value: the money value, isBonus: check whether this is the basic salary or bonus.

- **salarymanipulationlog**: The same collection of data from the salarymanipulation table, but every month all the records from the first table are moved to the second table to maintain performance.

- **studentclothes**: This collection keeps the clothes brought with the child to nursery with the date, UserID: the child, ClothesID: the clothe reference, Date: date of the day he brought this clothers.

- **studentrating**: This collection keeps the rating of the teacher on the student lesson, UserID: the child id, CurriculumID: the lesson id ,Rating: the rating grade, date: date of the rating.

- **timeslots**: This collection keeps the timeslots of the schedule, Begin: begining time, End: end time.

- **toilet**: This collection is a lookup table keeping the toilet types.

- **toiletcheck**: This collection keeps the records of the child toilet each day, TypeID: type of toilet, UserID: child id, Date: date of the day.

- **transaction**: This collection keeps the transactions of the nursery, UserID: the user doing the transaction, EventID: the event of this transaction, Date: date of transaction, Quantity: quantity of the transaction.

- **transactionlog**: The same collection of data from the transaction table, but every month all the records from the first table are moved to the second table to maintain performance.

- **user**: This collection keeps the records of the users, RoleID: the type of the user, UID: self referenece id of the parent user, DateAdded: the date the user is added to the system, status ID: the status of the user.

- **useractivities**: This collection keeps the records of the child activities, ActivityID: the id of the activity, the UserID: the child ID.

- **userclasscourse**: This collection assign each child to a class, UserID: child id, ClassID: class name.

- **userdailyhours**: This collection keeps the daily hours of the user, UserID: worker id, Date: date of the day, Hours: total hours, isExtra: did he worked extra hours.

- **userdailyhourslog**: The same collection of data from the userdailyhours table, but every month all the records from the first table are moved to the second table to maintain performance.

- **userstatus**: This collection keeps the user status which is customized by the admin.

- **uservaccination**: This collection keeps the childs which will take vaccinations, UserID: the child id, VaccinationID: the name of the vaccination.

- **vaccination**: This collection is a lookup table keeping the vaccinations.

- **word**: This collection keeps the titles words of every page, Name: the word itself, PageID: the page referencing this word.

- **wordlang**: This collection keeps the word and language for localization, WordID: the word name, LangID: the language of the word, Value: the value of the word in another language.

- **workershourssalary**: This collection keeps the salary per hour of each role type, BasicHour: the price of the basic hour, ExtraHour: the price of extra hour, DeducationHour: the prince of deduction hour, RoleID: the role name, NormalHours: normal working hours of this role.

5 Component Design

Not available

6 Humnan Interface Design

6.1 Overview of User Interface

The system interface will be divided into three different interfaces.

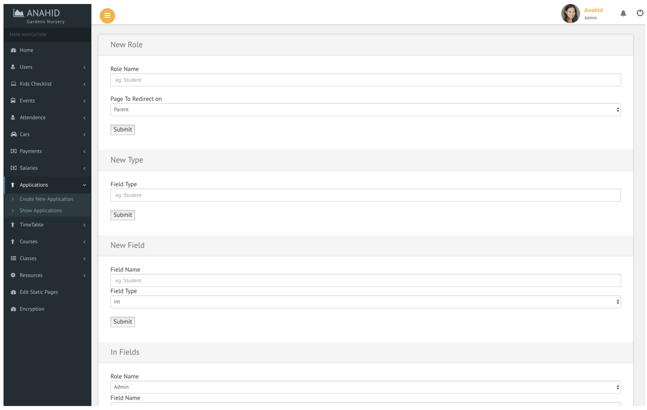
- **Admin interface** in which he can control all the functionalities, once he/she login, a navigation bar with all the system options will be showed.

- **Teacher interface** in which he/she can see their personal schedule and give grades to the course.

- **Child interface** in which he/she will see his daily report.

6.2 Screen Images

Here are some screenshots of the web application



New Role

Role Name: eg: Student

Page To Redirect on: Parent

New Type

Field Type: eg: Student

New Field

Field Name: eg: Student

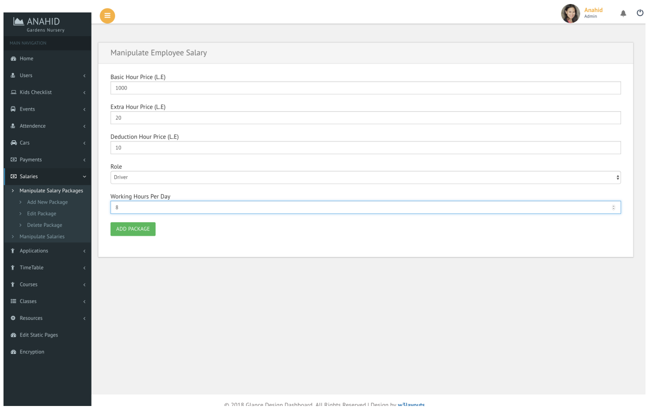
Field Type: int

In Fields

Role Name: Admin

Field Name

Figure 12: Create application web page



Manipulate Employee Salary

Basic Hour Price (L.E): 1000

Extra Hour Price (L.E): 20

Deduction Hour Price (L.E): 10

Role: Driver

Working Hours Per Day: 8

Figure 13: Create Salary Package web page

6.3 Screen Objects and Actions

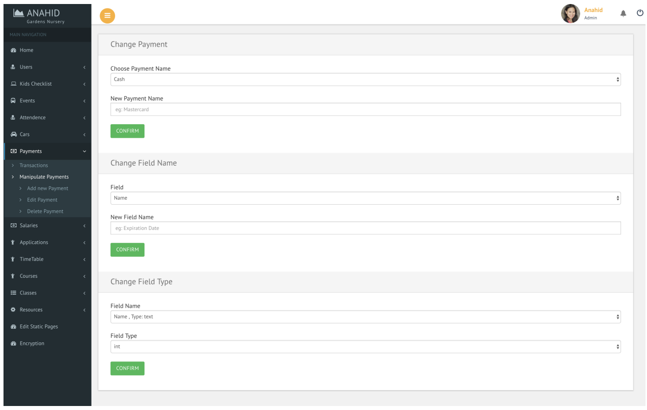
A discussion of screen objects and actions associated with those objects.

6.3.1 Show users

admin can see the available users of the system and show their status and if they don't have username or password it will show a red button saying 'missing login' which will redirect to page to enter username and password.

- **Admin Interface**

7 Requirements Matrix



Change Payment

Choose Payment Name: Cash

New Payment Name: eg: Mastercard

Change Field Name

Field: Name

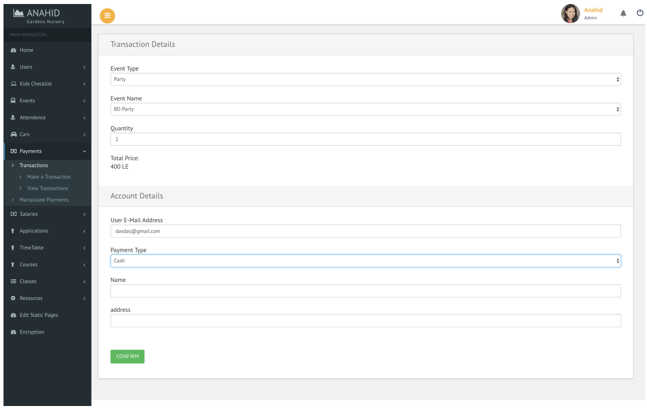
New Field Name: eg: Expiration Date

Change Field Type

Field Name: Name , Type: text

Field Type: int

Figure 14: Edit Payment web page



Transaction Details

Event Type: Party

Event Name: BD Party

Quantity: 2

Total Price:
400 LE

Account Details

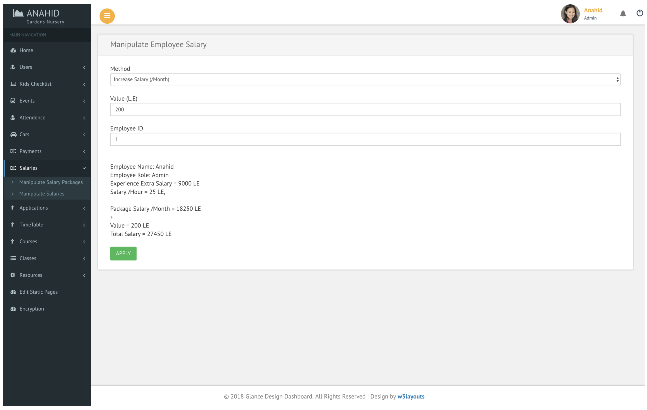
User E-Mail Address: dasdas@gmail.com

Payment Type: Cash

Name

address

Figure 15: Make transaction web page



Manipulate Employee Salary

Method: Increase Salary (/Month)

Value (L.E): 200

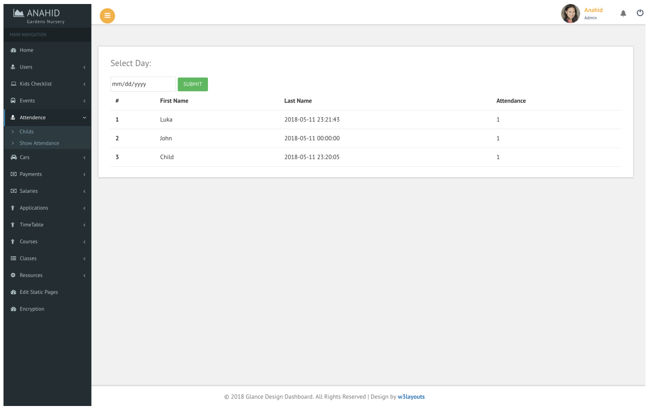
Employee ID: 1

Employee Name: Anahid
Employee Role: Admin
Experience Extra Salary = 9000 LE
Salary /Hour = 25 LE,

Package Salary /Month = 18250 LE
+
Value = 200 LE
Total Salary = 27450 LE

© 2018 Glance Design Dashboard. All Rights Reserved | Design by w3layouts

Figure 16: Manipulate Employee Salary web page



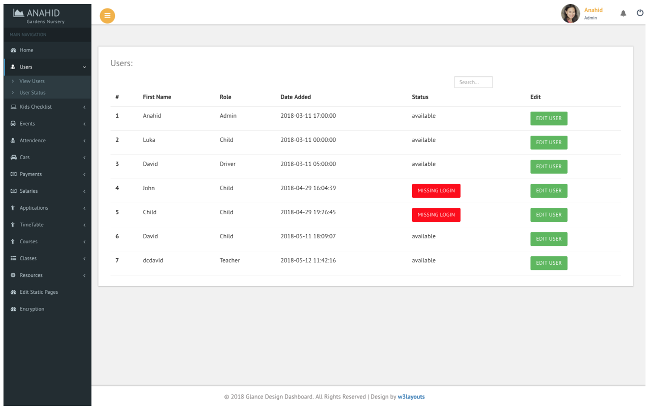
Select Day:

mm/dd/yyyy

#	First Name	Last Name	Attendance
1	Luka	2018-05-11 23:21:43	1
2	John	2018-05-11 00:00:00	1
3	Child	2018-05-11 23:20:05	1

© 2018 Glance Design Dashboard. All Rights Reserved | Design by w3layouts

Figure 17: View attendance web page

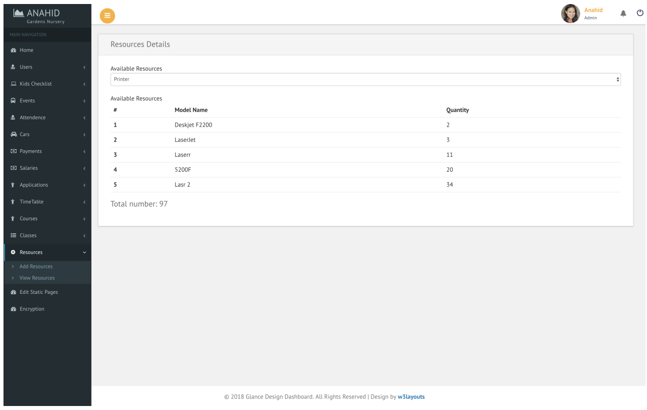


Users:

#	First Name	Role	Date Added	Status	Edit
1	Anahid	Admin	2018-03-11 17:00:00	available	EDIT USER
2	Luka	Child	2018-03-11 00:00:00	available	EDIT USER
3	David	Driver	2018-03-11 05:00:00	available	EDIT USER
4	John	Child	2018-04-29 16:04:39	MISSING LOGIN	EDIT USER
5	Child	Child	2018-04-29 19:26:45	MISSING LOGIN	EDIT USER
6	David	Child	2018-05-11 18:09:07	available	EDIT USER
7	dcdavid	Teacher	2018-05-12 11:42:16	available	EDIT USER

© 2018 Glance Design Dashboard. All Rights Reserved | Design by w3layouts

Figure 18: View users web page



Resources Details

Available Resources: Printer

Available Resources

#	Model Name	Quantity
1	Deskjet F2200	2
2	LaserJet	3
3	Laserr	11
4	5200F	20
5	Lasr 2	34

Total number: 97

© 2018 Glance Design Dashboard. All Rights Reserved | Design by w3layouts

Figure 19: View resources web page

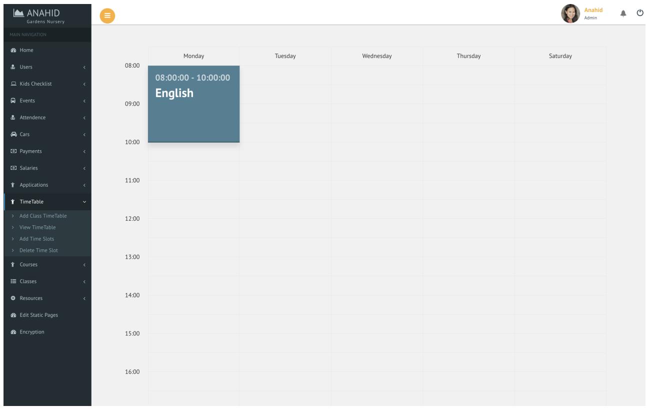


Figure 20: View Timetable web page

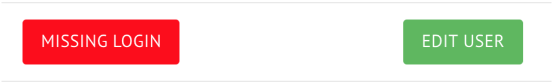
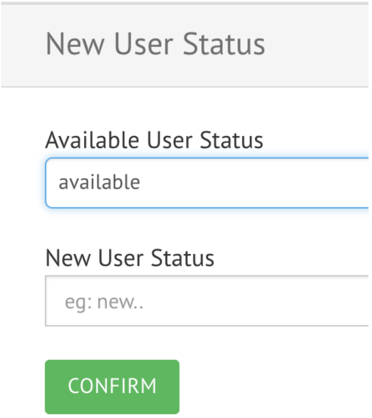


Figure 21: 'Missing login' to add username and password for users without username or password, 'Edit User' to edit user data

Figure 22: 'Search' search for users



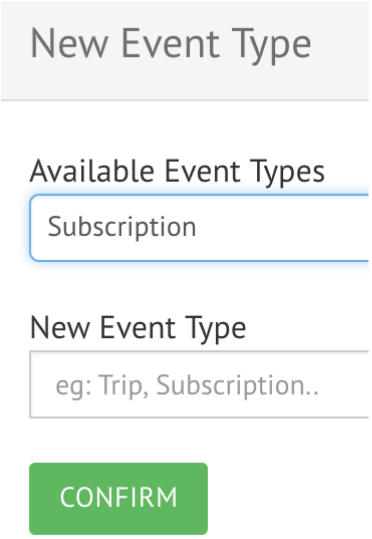
New User Status

Available User Status

available

New User Status

Figure 23: 'Confirm' add new user status



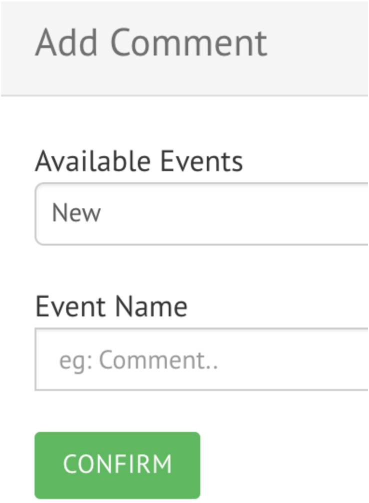
New Event Type

Available Event Types

Subscription

New Event Type

Figure 24: 'Confirm' add new event type



Add Comment

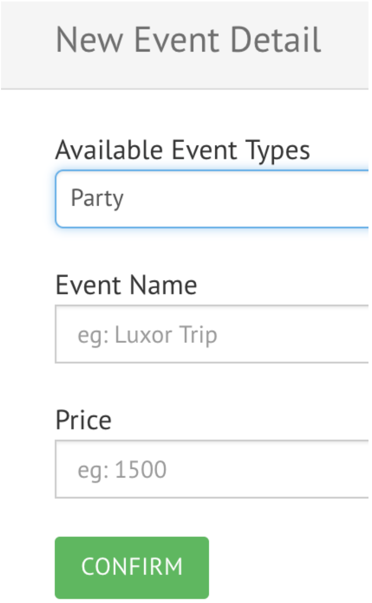
Available Events

New

Event Name

eg: Comment..

Figure 25: ’Confirm’ add new comment



New Event Detail

Available Event Types

Party

Event Name

eg: Luxor Trip

Price

eg: 1500

Figure 26: ’Confirm’ add new event detail

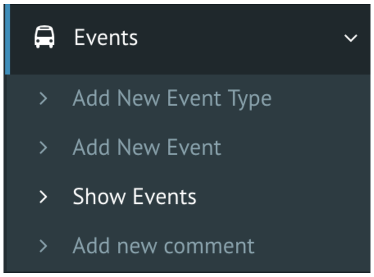
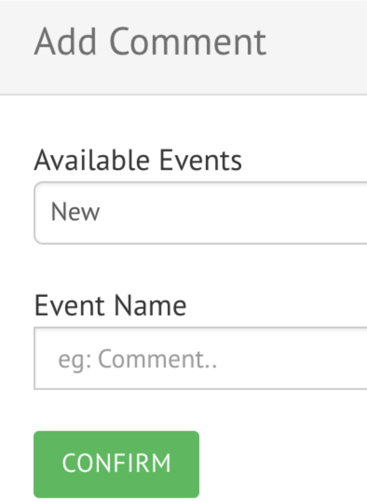


Figure 27: ’Show Events’



Add Comment

Available Events
New

Event Name
eg: Comment..

Figure 28: ’Confirm’ add new comment

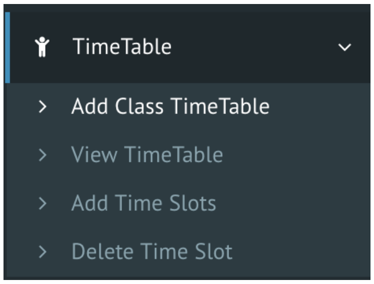


Figure 29: ’Add Class Timetable’ add timetable to a class

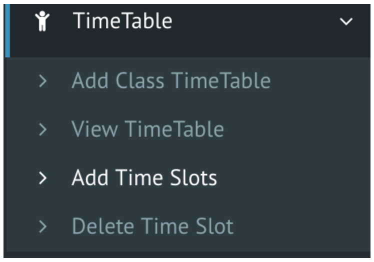


Figure 30: ’Add Timeslot’ add new timeslot for the schedule

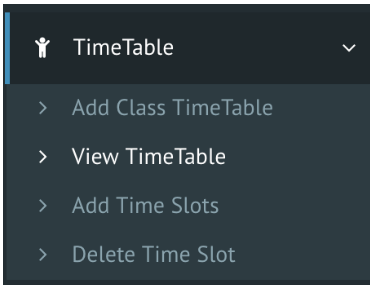


Figure 31: ’View Timetable’ view timetable for selected class

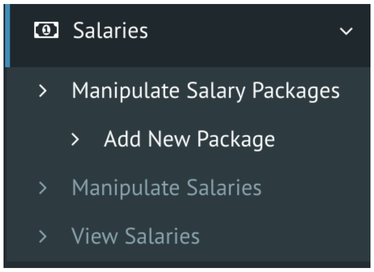
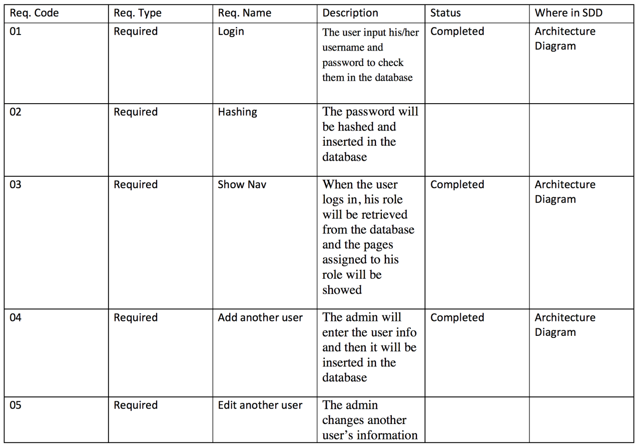
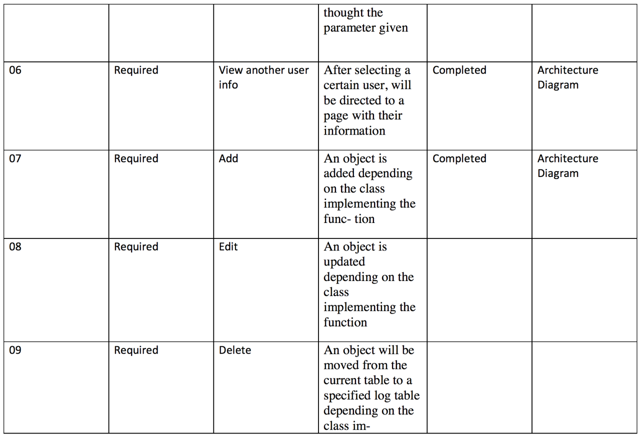


Figure 32: ’Add new package’ add new salary package for the workers



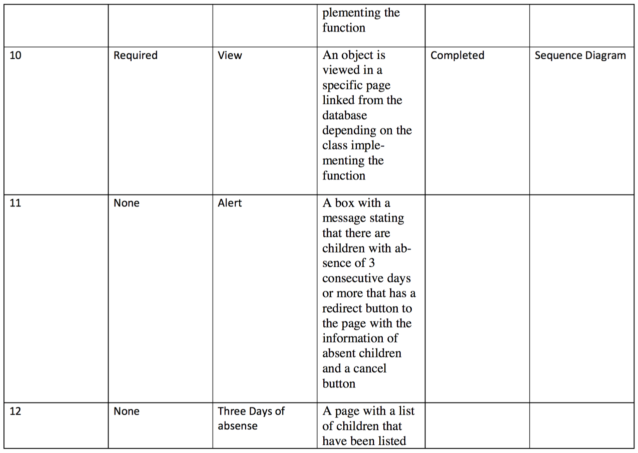
Req. Code	Req. Type	Req. Name	Description	Status	Where in SDD
01	Required	Login	The user input his/her username and password to check them in the database	Completed	Architecture Diagram
02	Required	Hashing	The password will be hashed and inserted in the database		
03	Required	Show Nav	When the user logs in, his role will be retrieved from the database and the pages assigned to his role will be showed	Completed	Architecture Diagram
04	Required	Add another user	The admin will enter the user info and then it will be inserted in the database	Completed	Architecture Diagram
05	Required	Edit another user	The admin changes another user's information		

Figure 33: Requirements Matrix



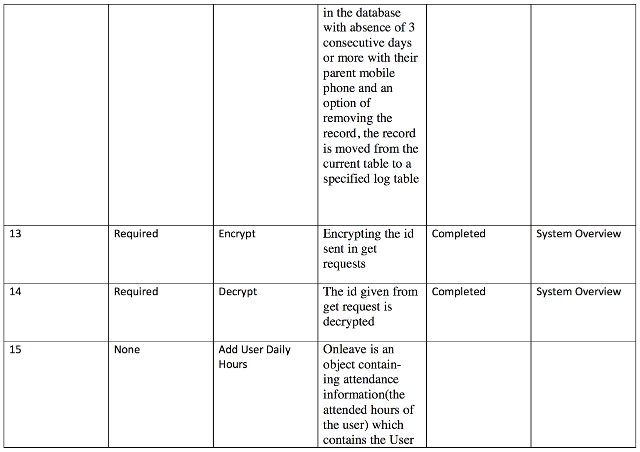
			thought the parameter given		
06	Required	View another user info	After selecting a certain user, will be directed to a page with their information	Completed	Architecture Diagram
07	Required	Add	An object is added depending on the class implementing the func- tion	Completed	Architecture Diagram
08	Required	Edit	An object is updated depending on the class implementing the function		
09	Required	Delete	An object will be moved from the current table to a specified log table depending on the class im-		

Figure 34: Requirements Matrix



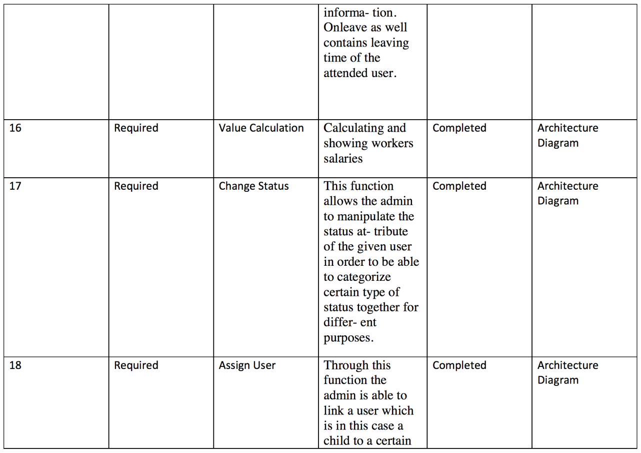
			plementing the function		
10	Required	View	An object is viewed in a specific page linked from the database depending on the class implementing the function	Completed	Sequence Diagram
11	None	Alert	A box with a message stating that there are children with absence of 3 consecutive days or more that has a redirect button to the page with the information of absent children and a cancel button		
12	None	Three Days of absense	A page with a list of children that have been listed		

Figure 35: Requirements Matrix



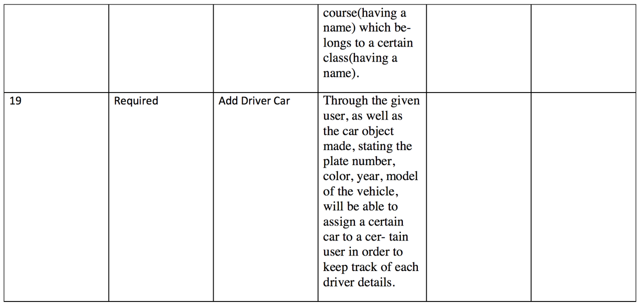
			in the database with absence of 3 consecutive days or more with their parent mobile phone and an option of removing the record, the record is moved from the current table to a specified log table		
13	Required	Encrypt	Encrypting the id sent in get requests	Completed	System Overview
14	Required	Decrypt	The id given from get request is decrypted	Completed	System Overview
15	None	Add User Daily Hours	Onleave is an object containing attendance information(the attended hours of the user) which contains the User		

Figure 36: Requirements Matrix



			informa- tion. Onleave as well contains leaving time of the attended user.		
16	Required	Value Calculation	Calculating and showing workers salaries	Completed	Architecture Diagram
17	Required	Change Status	This function allows the admin to manipulate the status at- tribute of the given user in order to be able to categorize certain type of status together for differ- ent purposes.	Completed	Architecture Diagram
18	Required	Assign User	Through this function the admin is able to link a user which is in this case a child to a certain	Completed	Architecture Diagram

Figure 37: Requirements Matrix



			course(having a name) which be- longs to a certain class(having a name).		
19	Required	Add Driver Car	Through the given user, as well as the car object made, stating the plate number, color, year, model of the vehicle, will be able to assign a certain car to a cer- tain user in order to keep track of each driver details.		

Figure 38: Requirements Matrix