**Back-end Training Plan**

**Duration: 3 months (November 2018 - January 2019)**

**Scope: Ruby & Rails**

**Prepared By**

* Ahmed Ba Haggag
* Ahmed Baslaib

**Goals**

* Get idea about Software Development The Big Picture
* Up and running with Ruby and Rails
* Evaluate students self learning capability and Passion
* Clickapps Culture

**Structure**

* Courses
* Homeworks
* Articles
* Assignments
* Followup
* Sessions

**Evaluation**

* Attendance
* Team Player
* Passion
* Proactive
* Self learning
* Homeworks
* Assignments
* Plan Follow Up

**Ruby Training Plan**

**Day 1: 1/11/2018**

* Welcome and Introduce Clickapps
* Introducing Training
  + Goals
  + Structure and materials
  + Courses
  + General Sessions
* Pereare system Ubuntu and required softwares
* Getting started with Ruby
  + Ruby Installation
  + Editor
  + How to run ruby program
* Ruby Data Types
  + Objects
  + Variables
  + Integers
  + Float
  + Strings
  + Symbols
  + Booleans
  + Ranges
  + Constants

**Weekend 2-3/11/2018**

* Programming Foundation: Fundamentals ***‘lynda course’***
* Value Type vs. Reference Type
* Difference between String and Symbol
* String Methods

**Day 2: 4/11/2018**

* Control Structures (conditions)
* if else elsif
* Unless
* Case
* *Homework*
  + Find net salary for employee when the bonus rate is as follows
    - basic salary between 0 and 10000 then bonus 25%.
    - basic salary between 10001 and 100000 then bonus 15%.
    - basic salary > 100000 then bonus 10%.

Print basic salary, bonus amount and net salary

**Day 3: 5/11/2018**

* Collections
  + Arrays
  + Hashes
* Control Structures (repetitive)
  + Loops
  + Iterators
* Programming Foundation: Data Structures (*lynda course*)
* Homework
  + Suggestion game
  + Write a Ruby program to retrieve the total marks where subject name and marks of a student stored in a hash. Sample subject and marks : Literature -74, Science – 89, Math-91

**Day 4: 6/11/2018**

* Code Blocks
  + Find
  + Merge
  + Collect
  + Sort
  + Inject
* Programming Foundation: Data Structures (*lynda course*)
* Homework

having array of employees:

**employees = [**

**{**

**id: 1,**

**name: 'Ahmed',**

**age: 20,**

**salary: [4000, 2000],**

**reported\_employees: [2, 3]**

**},**

**{**

**id: 2,**

**name: 'Ali',**

**age: 15,**

**salary: [2000],**

**reported\_employees: []**

**},**

**{**

**id: 3,**

**name: 'Salem',**

**age: 20,**

**salary: [4000],**

**reported\_employees: []**

**},**

**{**

**id: 4,**

**name: 'Fahd',**

**age: 30,**

**salary: [5000, 2000],**

**reported\_employees: [5]**

**},**

**{**

**id: 5,**

**name: 'Ahmed',**

**age: 18,**

**salary: [20000],**

**reported\_employees: []**

**},**

**]**

Q1: find all employees with salary greater than 7000

Q2: find employee with id = 2

Q3: Sort employees by highest salary

Q4: Get employees with these data only (:id, :name) and their all reported employees info

Q5: Get total salary for all employees

Q6: Add this key-value pair {is\_manager: true} for all employees having reported employees

**Day 5: 7/11/2018**

* Methods
  + Defining and calling method
  + Variable scope in methods
  + Argument
  + Argument default values
  + Return value
  + Operators also are methods
* Homework
  + Write a Ruby program to sum up all the numbers inside (use recursive function)

[1, 1, 1, [3, 4, [8]], [5]]

* + Give examples for all method arguments
* Articles
  + Pass by value vs. Pass by reference
  + Method Arguments
  + Overloading in Ruby
  + Recursive function

**Day 6: 8/11/2018 + Weekend 9-10/11/2018**

**Assignment 1**

You will build complete employees management app and will include all previous sections. Data required for each employee is as follows (hash keys): :emp\_id, :name, :joining\_date, :salary, :department

**{**

**emp\_id: 1,**

**name: 'Ahmed',**

**joining\_date: Date.new(2018,11,1),**

**department: 'back-end',**

**salary: 20000**

**}**

This assignment includes the following requirements:

* Create, update and delete employee
* Emp\_id id unique so don’t allow to add employee with existing employee emp\_id
* Searching by emp\_id and name
* Sorting
* Employees report, example

Emp ID, Name, Joining Date, Department, Salary

------------------------------------------------------------------

259, Ahmed Aboud Bahaggag, 1/1/2014, Back-end, 50000

260, Ahmed Omer Baslaib, 1/1/2017, Back-end, 60000

* Group employees by department and get total salaries for each department, example:

Department name, Total Salaries

--------------------------------------------

Back-end, 50000

Front-end, 35000

* Get first and last employee joined
* Get employee with low and high salary

**Day 7: 11/11/2018**

* Assignment deadline and discussion
* Review homeworks
* Prepare for next section

**Day 8: 12/11/2018**

* Classes
  + Defining and using classes
  + Instances
  + Attributes
  + reader/writer methods
  + Attribute methods
  + Initialize method

**Day 9: 13/11/2018**

* Classes
  + Class methods
  + Class attributes
  + Class reader /writer methods
  + Inheritance
  + Subclasses overriding
  + Accessing the super classes

**Day 10, 11: 14-15/11/2018**

* Classes: 101 OOP Session (Review)
* Modules
  + Namespacing
  + Mixins
  + Load require and include
  + Enumerable as mixins

**Weekend 16-17/11/2018**

* **Assignment 2:**

You will build employees CRUD operations use Employee Class. Each employee will have the following attributes:

* Id: which auto incremented integer
* Emp\_id: unique employee id
* Name
* Joining\_date: should be added as string but saved as Date object
* Salary

We will have the following operations:

* Create: which can be called as Employee.create(hash). Employee info should be passed into create method after it’s processed outside create method
* Save: which can be called as obj.save .. here we need first to initiate new employee object like this obj = Employee.new(hash) employee info should be managed same as create
* Update: which can be called as Employee.update(id, data: hash) here we should pass only data need to be updated. If you find empty value then ignore it and keep previous value
* Delete: which can be called as Employee.delete(id) or obj.delete
* List: here we will have 2 options: search and sort
  + Search: ask user if he want to search, if yes then ask him to enter the search term. Search will be implemented by emp\_id or name
  + Sort: ask user if he wants to sort, if yes then ask him to enter sort by criteria “emp\_id, name, joining\_date, salary”

After getting user inputs now you will implement listing according to user inputs for searching and sorting

* Programming Foundation: Object Oriented Design (*lynda course*)
* Exceptions and Error Handling in Ruby
* Date and DateTime classes and their methods
* Variables scope

**Day 12: 18/11/2018**

* Assignment deadline and discussion
* Prepare for next section

**Day 13: 19/11/2018**

* Working with Files
  + input/output basics
  + File system basics
  + File paths
  + Accessing files
  + Writing to files
  + Reading from files
  + File pointer
  + Renaming and deleting files
  + Examine file details
  + Working with directories

**Day 14: 20/11/2018**

* Ruby Project: Creating The Food Finder
  + Project overview
  + Application path
  + Guide class
  + Restaurant class
  + Accessing the restaurant file
  + Handling input in the action loop
  + Limiting input
  + Adding restaurants
  + Refactoring the add action
  + Listing restaurants
  + Improving output
  + Finding restaurants
  + Sorting

**Day 15, 16, 17: 21-22,25/11/2018, Weekend (23-24/11/2018)**

**Final Project TODO**

**Day 18, 19, 20, 21: 26-29/11/2018, Weekend (30/11/2018)**

**Sessions**

* The Big Picture Series (General Sessions):
  + Software Development
  + System Analysis and Development Methodologies
  + Back-end Development
  + Front-end Development
  + Mobile Development
  + QA Concepts
  + Cloud Concepts
  + DevOps Concepts