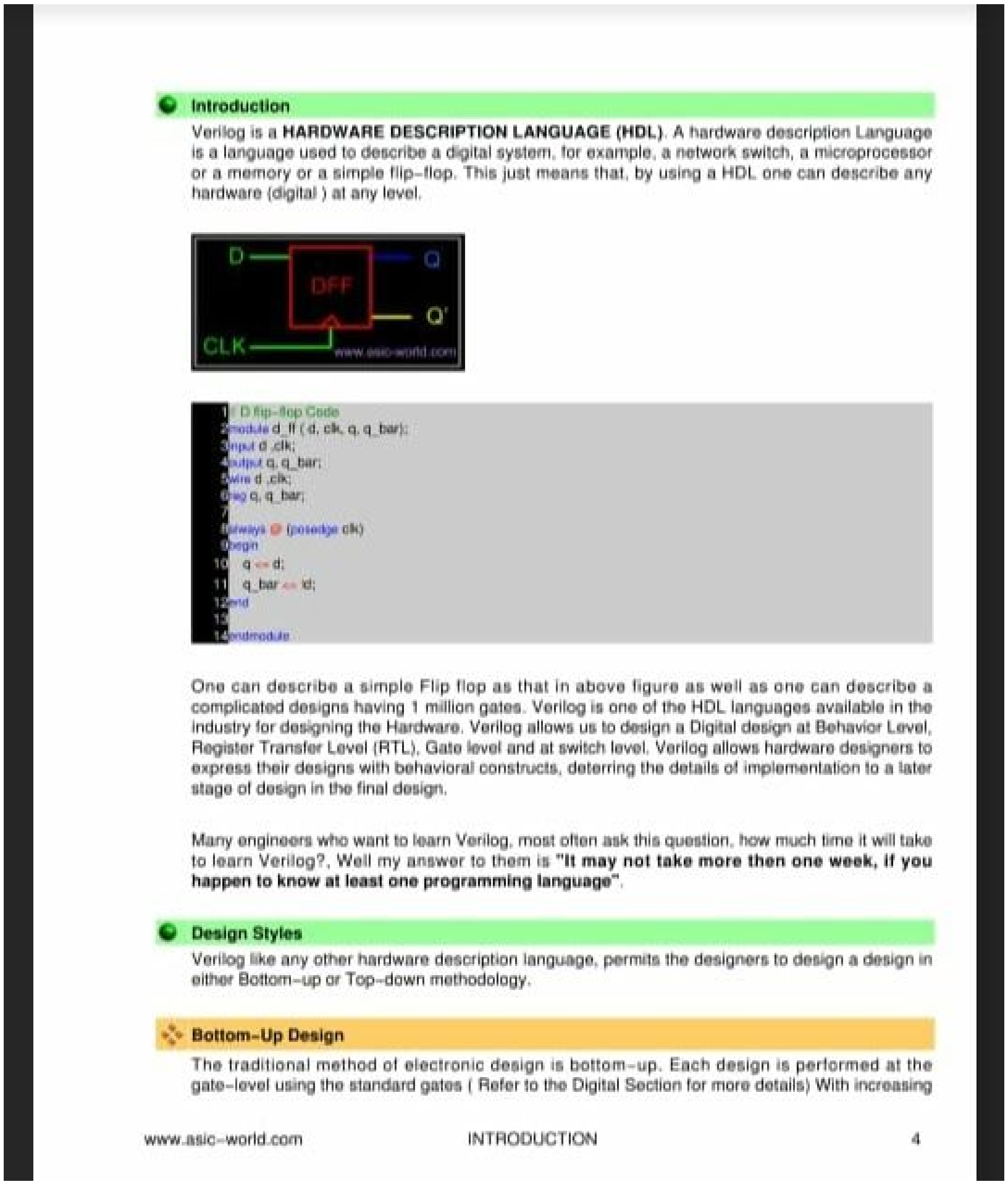
DAILY ASSESSMENT FORMAT

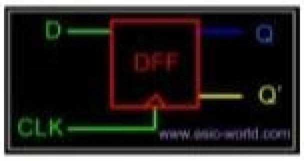
Date:	**5-06-2020**	**Name:**	**MOUNITHA D M**
Course:	**DIGITAL DESIGN USING HDL**	**USN:**	**4AL17EC055**
Topic:	**Verilog Tutorials and practiceprogram Building Demo projects using FGPA**	**Semester & Section:**	**6TH SEM "A" SEC**
Github Repository:	**Mounitha_-ec055**		

FORENOON SESSION DETAILS
Image of session



Introduction

Verilog is a **HARDWARE DESCRIPTION LANGUAGE (HDL)**. A hardware description Language is a language used to describe a digital system, for example, a network switch, a microprocessor or a memory or a simple flip–flop. This just means that, by using a HDL one can describe any hardware (digital) at any level.



```
// D flip-flop Code
module d_ff ( d, clk, q, q_bar);
input d ,clk;
output q, q_bar;
wire d ,clk;
reg q, q_bar;

always @ (posedge clk)
begin
  q <= d;
  q_bar <= !d;
end

endmodule
```

One can describe a simple Flip flop as that in above figure as well as one can describe a complicated designs having 1 million gates. Verilog is one of the HDL languages available in the industry for designing the Hardware. Verilog allows us to design a Digital design at Behavior Level, Register Transfer Level (RTL), Gate level and at switch level. Verilog allows hardware designers to express their designs with behavioral constructs, deterring the details of implementation to a later stage of design in the final design.

Many engineers who want to learn Verilog, most often ask this question, how much time it will take to learn Verilog?, Well my answer to them is **"It may not take more then one week, if you happen to know at least one programming language"**.

Design Styles

Verilog like any other hardware description language, permits the designers to design a design in either Bottom–up or Top–down methodology.

Bottom–Up Design

The traditional method of electronic design is bottom–up. Each design is performed at the gate–level using the standard gates (Refer to the Digital Section for more details) With increasing

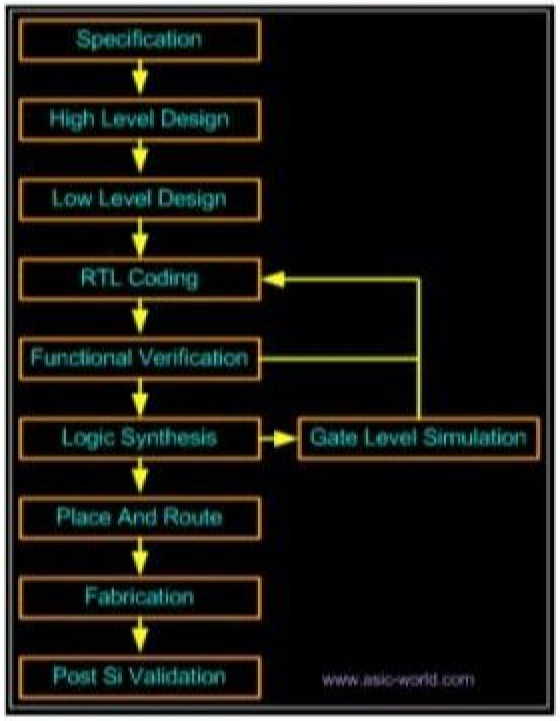
www.asic–world.com INTRODUCTION 4

complexity of new designs this approach is nearly impossible to maintain. New systems consi
ASIC or microprocessors with a complexity of thousands of transistors. These traditio
bottom-up designs have to give way to new structural, hierarchical design methods. Without th
new design practices it would be impossible to handle the new complexity.

Top-Down Design

The desired design-style of all designers is the top-down design. A real top-down design all
early testing, easy change of different technologies, a structured system design and offers m
other advantages. But it is very difficult to follow a pure top-down design. Due to this fact m
designs are mix of both the methods, implementing some key elements of both design styles.

Figure shows a Top-Down design approach.

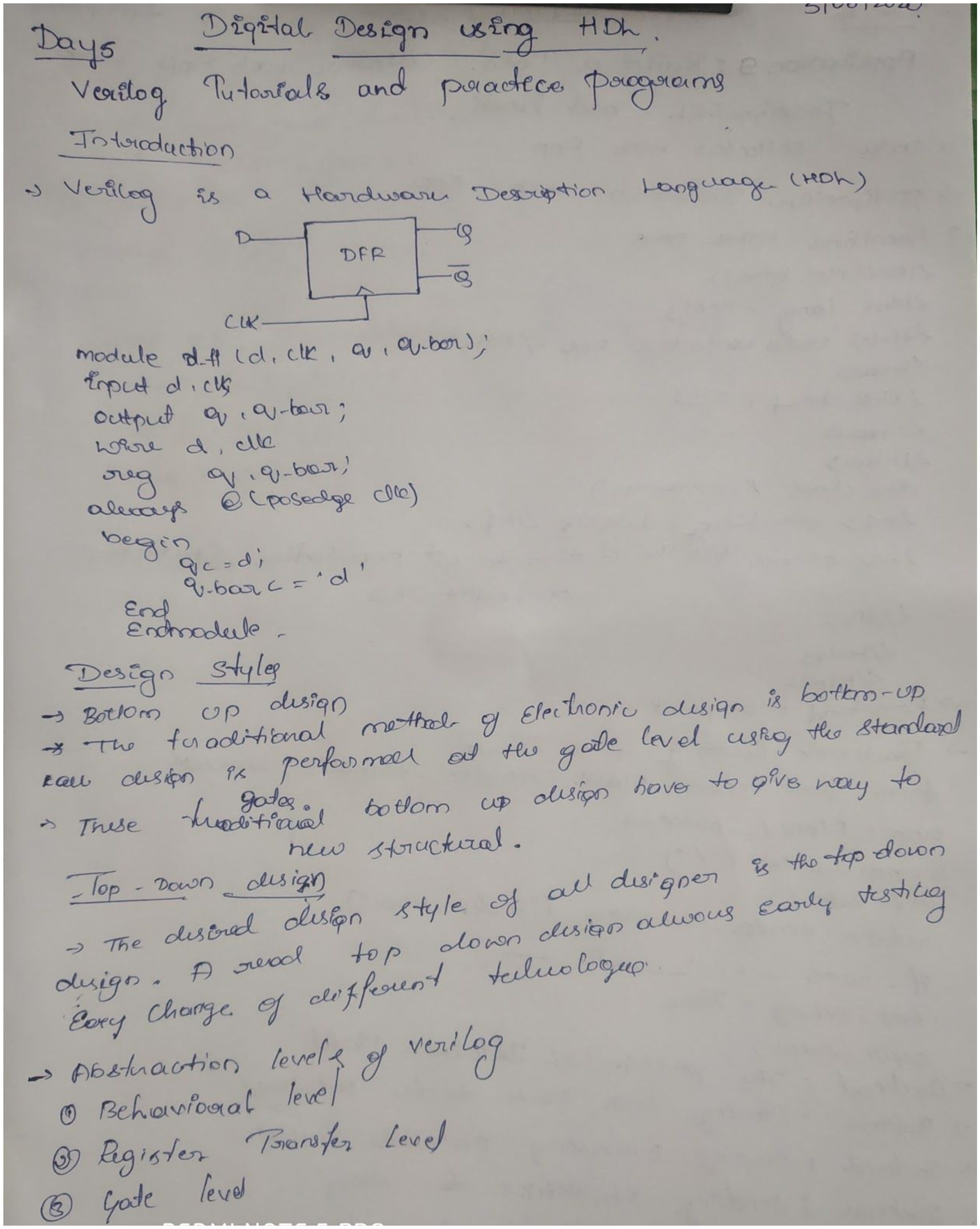


Abstraction Levels of Verilog

Verilog supports a design at many different levels of abstraction. Three of them are very import

- Behavioral level

Report – Report can be typed or hand written for up to two pages.



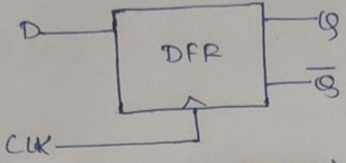
5/08/202

Day5 Digital Design using HDL.

Verilog Tutorials and practice programs

Introduction

→ Verilog is a Hardware Description Language (HDL)



```
module d-ff (d, clk, q, q-bar);
 input d, clk
  output q, q-bar;
  wire d, clk
  reg  q, q-bar;
 always @(posedge clk)
  begin
       q<=d;
       q-bar<='d'
    End
    Endmodule.
```

Design Styles

→ Bottom up design

→ The traditional method of Electronic design is bottom-up cell design is performed at the gate level using the standard gates.

→ These ~~traditional~~ bottom up design have to give way to new structural.

Top - Down design

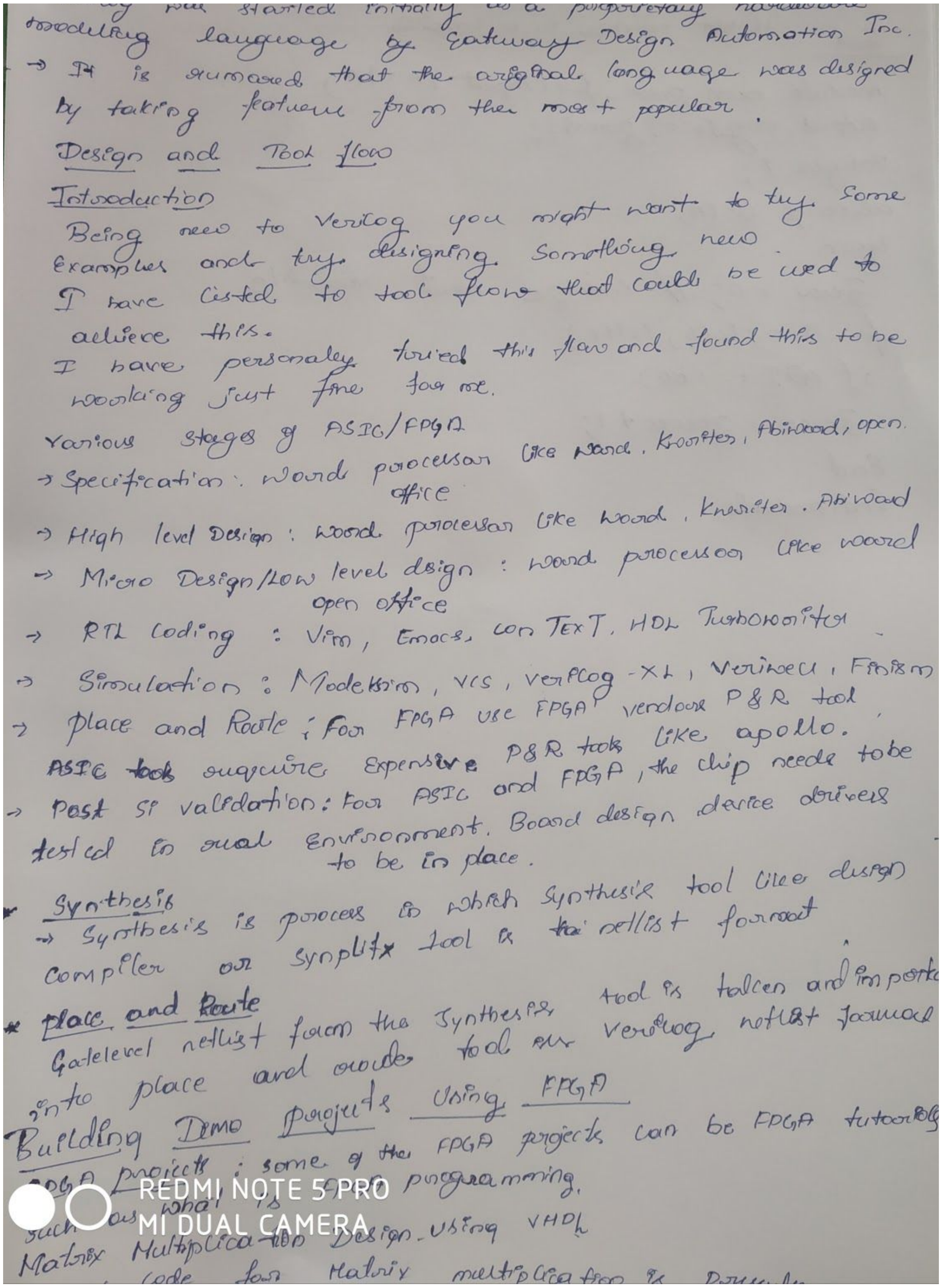
→ The desired design style of all designer is the top down design. A read top down design allows early testing Easy change of different technologies

→ Abstraction levels of verilog

① Behavioural level

② Register Transfer Level

③ Gate level



[illegible] was started initially as a proprietary hardware modelling language by Gateway Design Automation Inc.

→ It is rumored that the original language was designed by taking features from the most popular.

Design and Tool flow

Introduction

Being new to Verilog you might want to try some examples and try designing something new. I have listed to tool flow that could be used to achieve this.

I have personaly tried this flow and found this to be working just fine for me.

Various stages of ASIC/FPGA

→ Specification: Word processor like Word, Kwriter, Abiword, open office

→ High level Design: Word processor like Word, Kwriter, Abiword

→ Micro Design/Low level dsign: Word processor like Word open office

→ RTL Coding: Vim, Emacs, conTEXT, HDL Turbowriter

→ Simulation: Modelsim, VCS, Verilog-XL, Veriwell, Finsim

→ Place and Route: For FPGA use FPGA vendor P&R tool, ASIC tools require Expensive P&R tools like apollo.

→ Post SI validation: For ASIC and FPGA, the chip needs to be tested in real Environment. Board design device drivers to be in place.

* Synthesis

→ Synthesis is process in which synthesis tool like design compiler or synplify tool is the netlist format

* Place and Route

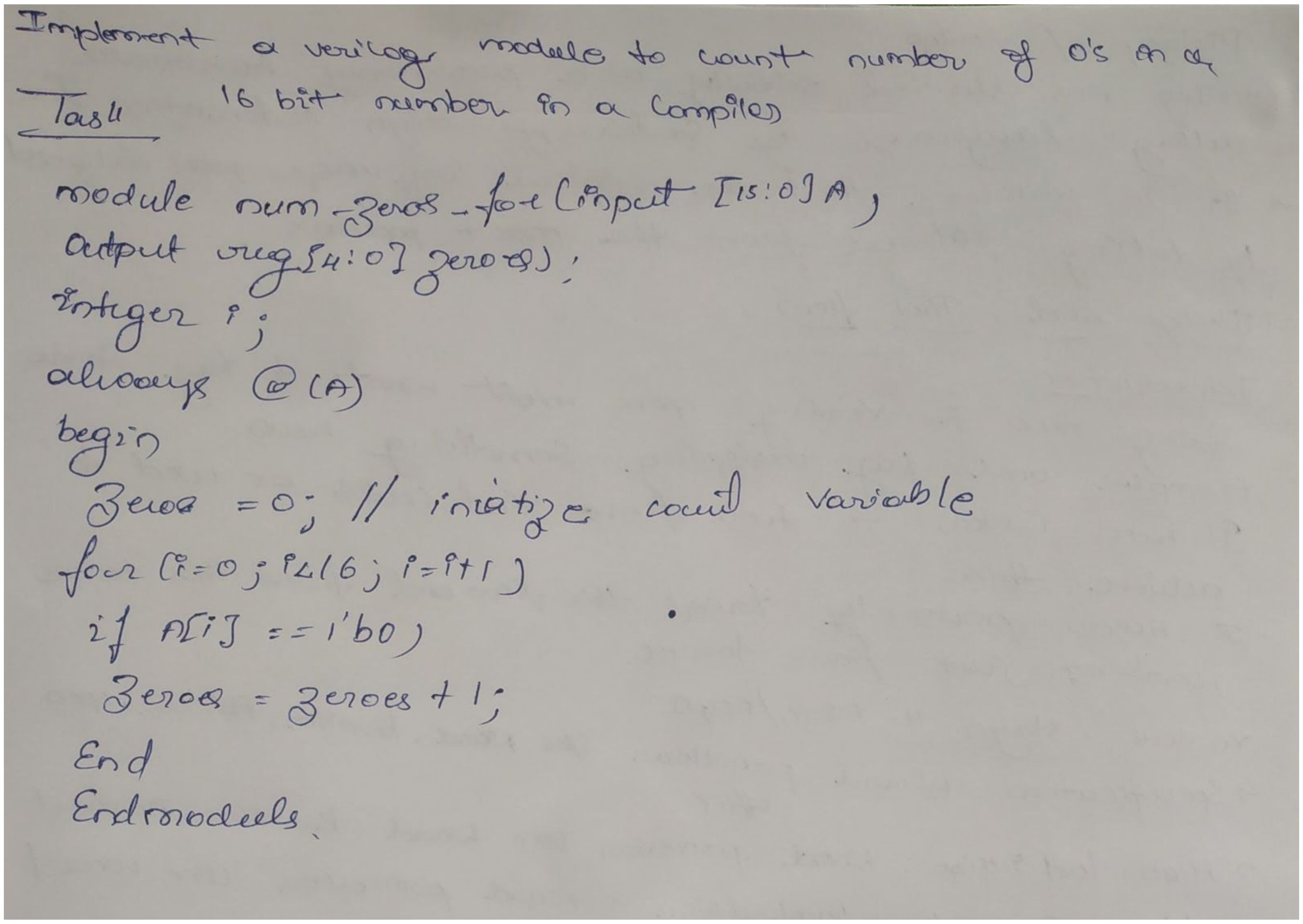
Gatelevel netlist from the synthesis tool is taken and imported into place and route tool in verilog netlist format

Building Demo projects Using FPGA

FPGA projects: some of the FPGA projects can be FPGA tutorial such as what is FPGA programming.

Matrix Multiplication Design Using VHDL

[illegible] code for Matrix multiplication is [illegible]



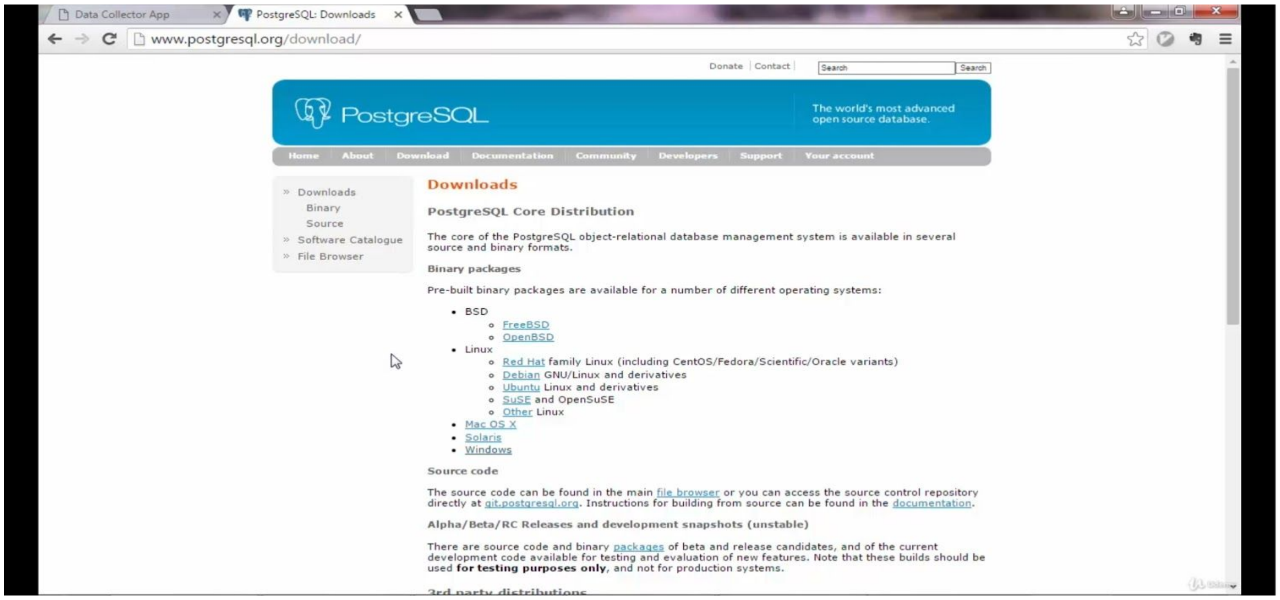
Implement a verilog module to count number of 0's in a
Task 16 bit number in a Compiler

```
module num_zeros_for (input [15:0] A,
output reg [4:0] zeroes);
integer i;
always @(A)
begin
 Zeros = 0; // iniatize count variable
for (i=0; i<16; i=i+1)
 if (A[i] == 1'b0)
  Zeroes = zeroes + 1;
End
Endmodels.
```

DATE	5-06-2020	Name:	MOUNITHA DM
Course:	PYTHON	USN:	4AL17EC055
Topic:	Application 9:Build a data collector web app with postgre SQL AND Flask	Semester & Section:	6TH SEM "A" SEC

AFTERNOON SESSION DETAILS

Image of session


Data Collector App
PostgreSQL: Downloads
www.postgresql.org/download/
Donate | Contact | Search
PostgreSQL
The world's most advanced open source database.
Home About Download Documentation Community Developers Support Your account
» Downloads
Binary
Source
» Software Catalogue
» File Browser

Downloads

PostgreSQL Core Distribution

The core of the PostgreSQL object-relational database management system is available in several source and binary formats.

Binary packages

Pre-built binary packages are available for a number of different operating systems:

- BSD
  - FreeBSD
  - OpenBSD
- Linux
  - Red Hat family Linux (including CentOS/Fedora/Scientific/Oracle variants)
  - Debian GNU/Linux and derivatives
  - Ubuntu Linux and derivatives
  - SuSE and OpenSuSE
  - Other Linux
- Mac OS X
- Solaris
- Windows

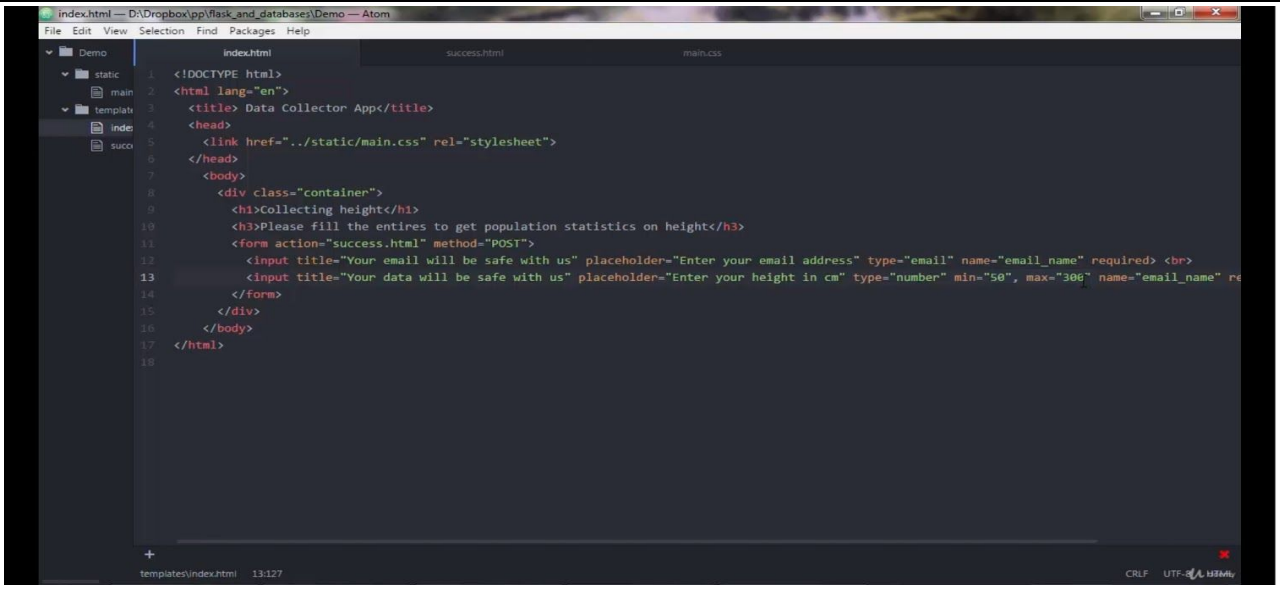
Source code

The source code can be found in the main file browser or you can access the source control repository directly at git.postgresql.org. Instructions for building from source can be found in the documentation.

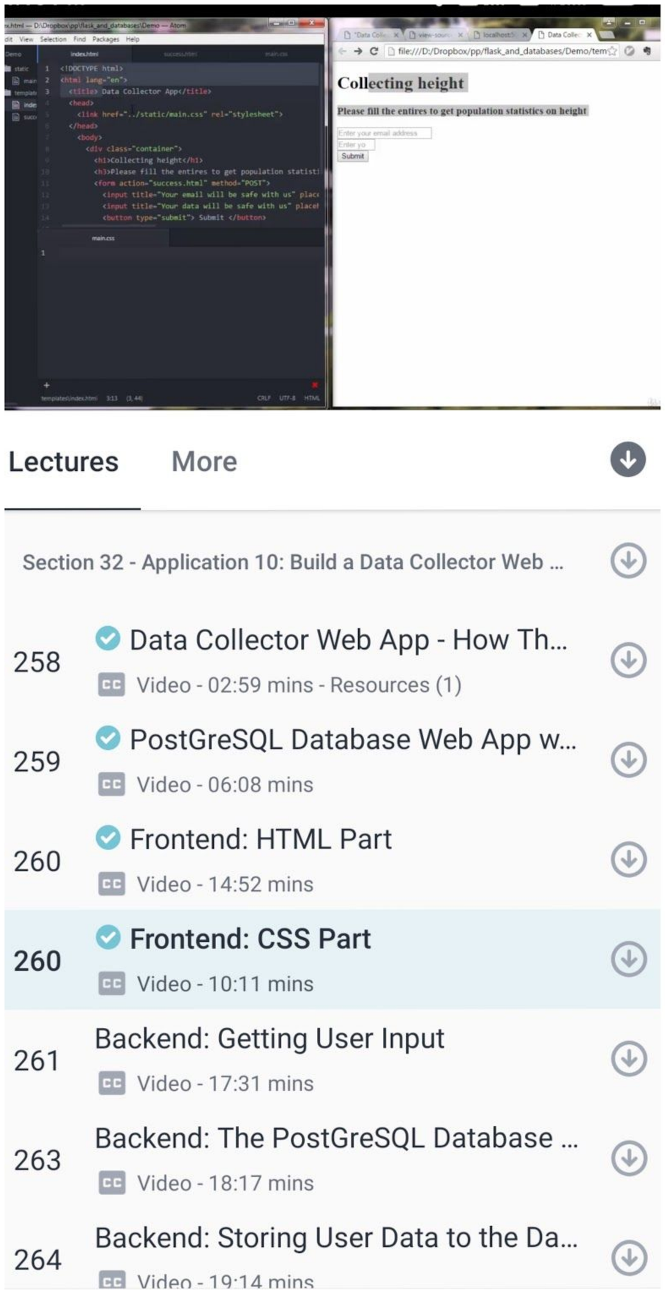
Alpha/Beta/RC Releases and development snapshots (unstable)

There are source code and binary packages of beta and release candidates, and of the current development code available for testing and evaluation of new features. Note that these builds should be used **for testing purposes only**, and not for production systems.

3rd party distributions


index.html — D:\Dropbox\pp\flask_and_databases\Demo — Atom
File Edit View Selection Find Packages Help
Demo
static
main
template
index
succ
index.html
success.html
main.css
templates\index.html 13:127
CRLF

```html
<!DOCTYPE html>
<html lang="en">
  <title> Data Collector App</title>
  <head>
    <link href="../static/main.css" rel="stylesheet">
  </head>
    <body>
      <div class="container">
        <h1>Collecting height</h1>
        <h3>Please fill the entires to get population statistics on height</h3>
        <form action="success.html" method="POST">
          <input title="Your email will be safe with us" placeholder="Enter your email address" type="email" name="email_name" required> <br>
          <input title="Your data will be safe with us" placeholder="Enter your height in cm" type="number" min="50", max="300" name="email_name" re
        </form>
      </div>
    </body>
</html>
```



```html
<!DOCTYPE html>
<html lang="en">
  <title> Data Collector App</title>
  <head>
    <link href="../static/main.css" rel="stylesheet">
  </head>
    <body>
      <div class="container">
        <h1>Collecting height</h1>
        <h3>Please fill the entires to get population statisti
        <form action="success.html" method="POST">
          <input title="Your email will be safe with us" plac
          <input title="Your data will be safe with us" placel
          <button type="submit"> Submit </button>
```

Collecting height

Please fill the entires to get population statistics on height

Enter your email address

Enter yo

Submit

Lectures More

Section 32 - Application 10: Build a Data Collector Web ...

258 Data Collector Web App - How Th...
Video - 02:59 mins - Resources (1)

259 PostGreSQL Database Web App w...
Video - 06:08 mins

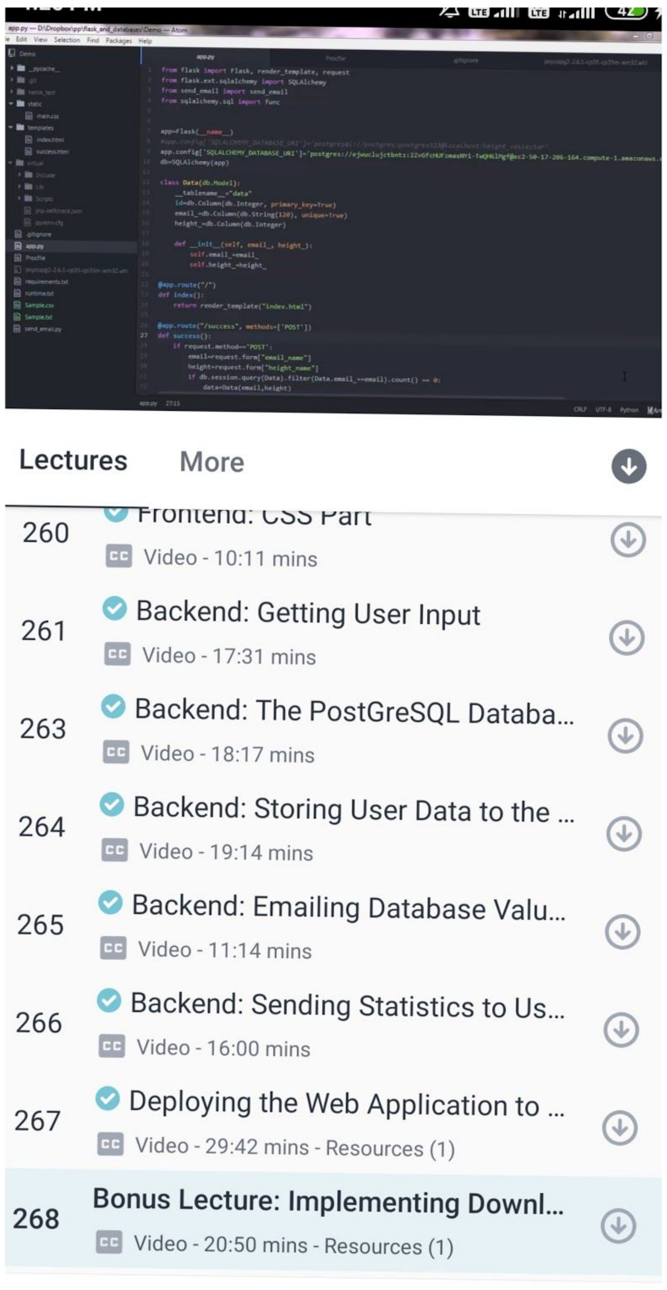
260 Frontend: HTML Part
Video - 14:52 mins

260 **Frontend: CSS Part**
Video - 10:11 mins

261 Backend: Getting User Input
Video - 17:31 mins

263 Backend: The PostGreSQL Database ...
Video - 18:17 mins

264 Backend: Storing User Data to the Da...
Video - 19:14 mins



```python
from flask import Flask, render_template, request
from flask.ext.sqlalchemy import SQLAlchemy
from send_email import send_email
from sqlalchemy.sql import func


app=Flask(__name__)
#app.config['SQLALCHEMY_DATABASE_URI']='postgresql://postgres:postgres123@localhost/height_collector'
app.config['SQLALCHEMY_DATABASE_URI']='postgres://ejwuclujctbntz:22vGfcHUFzmasNYi-TwQH6lMgf@ec2-50-17-206-164.compute-1.amazonaws.
db=SQLAlchemy(app)

class Data(db.Model):
    __tablename__="data"
    id=db.Column(db.Integer, primary_key=True)
    email_=db.Column(db.String(120), unique=True)
    height_=db.Column(db.Integer)

    def __init__(self, email_, height_):
        self.email_=email_
        self.height_=height_

@app.route("/")
def index():
    return render_template("index.html")

@app.route("/success", methods=['POST'])
def success():
    if request.method=='POST':
        email=request.form["email_name"]
        height=request.form["height_name"]
        if db.session.query(Data).filter(Data.email_==email).count() == 0:
            data=Data(email,height)
```

Lectures More

260 Frontend: CSS Part
Video - 10:11 mins

261 Backend: Getting User Input
Video - 17:31 mins

263 Backend: The PostGreSQL Databa...
Video - 18:17 mins

264 Backend: Storing User Data to the ...
Video - 19:14 mins

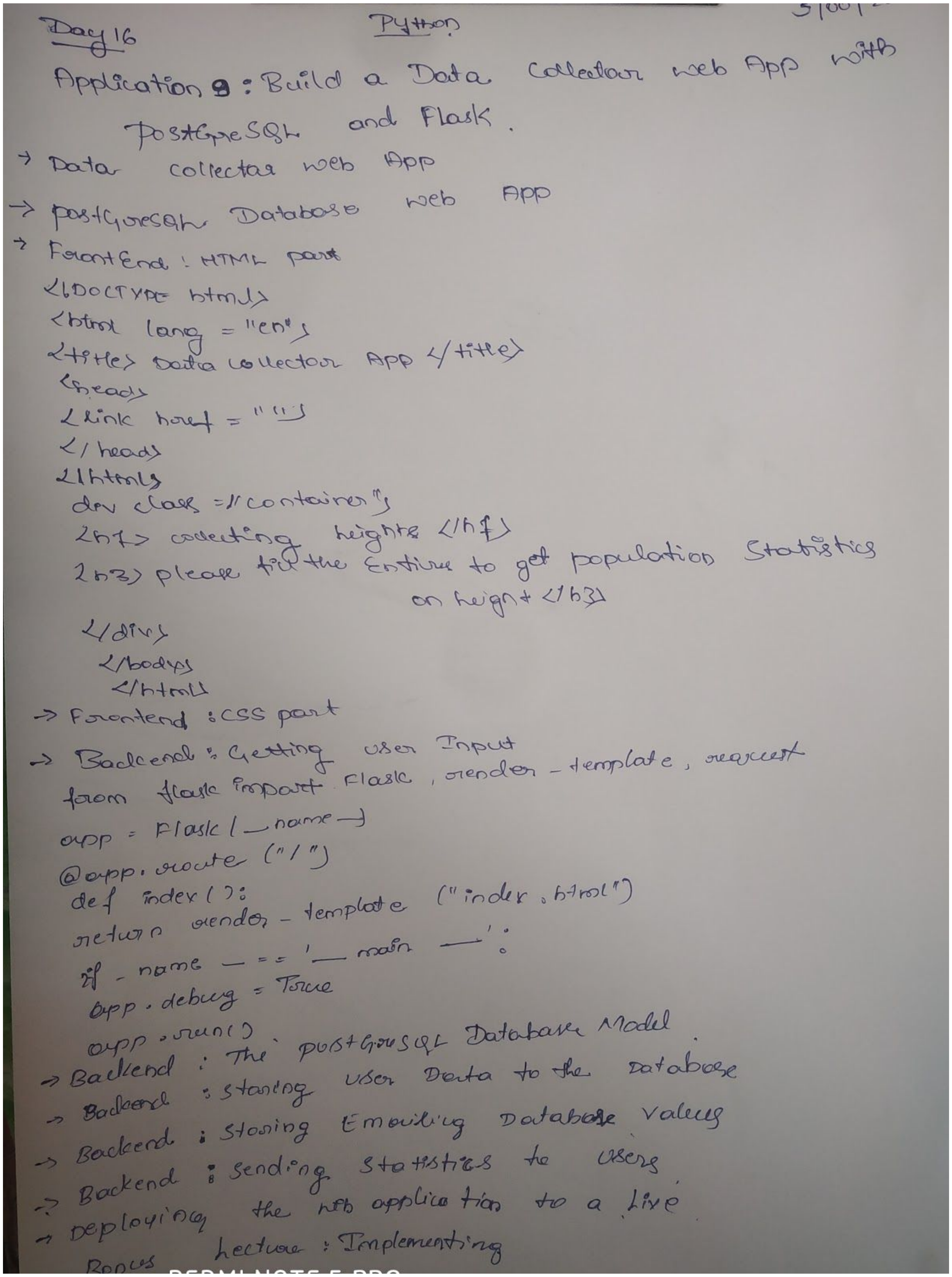
265 Backend: Emailing Database Valu...
Video - 11:14 mins

266 Backend: Sending Statistics to Us...
Video - 16:00 mins

267 Deploying the Web Application to ...
Video - 29:42 mins - Resources (1)

268 **Bonus Lecture: Implementing Downl...**
Video - 20:50 mins - Resources (1)

Report – Report can be typed or hand written for up to two pages.



Day 16 Python 5/06/2

Application 9 : Build a Data Collector web App with PostGreSQL and Flask.

→ Data collector web App

→ postGreSQL Database web App

→ FrontEnd : HTML part

```
<!DOCTYPE html>
<html lang = "en">
<title> Data Collector App </title>
<head>
<link href = "">
</head>
<html>
 div class = "container">
 <h1> collecting heights </h1>
 <h3> please fill the Entires to get population Statistics
                     on height </h3>
 </div>
  </body>
   </html>
```

→ Frontend : CSS part

→ Backend : Getting User Input

```
from flask import Flask , render - template , request
app = Flask ( _ name _ )
@app. route ("/")
 def index ( ):
 return render - template ("index . html")
 if _ name _ = = '_ main _':
  app . debug = True
  app . run()
```

→ Backend : The postGreSQL Database Model.

→ Backend : storing user Data to the Database

→ Backend : Storing Emailing Database Values

→ Backend : sending statistics to users

→ Deploying the web application to a Live

Bonus lecture : Implementing

REDMI NOTE 5 PRO