REPORT JUNE 04

|  |  |  |  |
| --- | --- | --- | --- |
| **Date:** | **04-06-2020** | **Name:** | **BHOOMIKA R HEBBAR** |
| **Course:** | **Digital Design Using HDL** | **USN:** | **4AL17EC010** |
| **Topic:** | **Hardware Modeling using**  **Verilog,Implement T Flip-Flop** | **Semester**  **& Section:** | **6th SEM A** |
| **Github**  **Repository:** | **bhoomika\_python** |  |  |

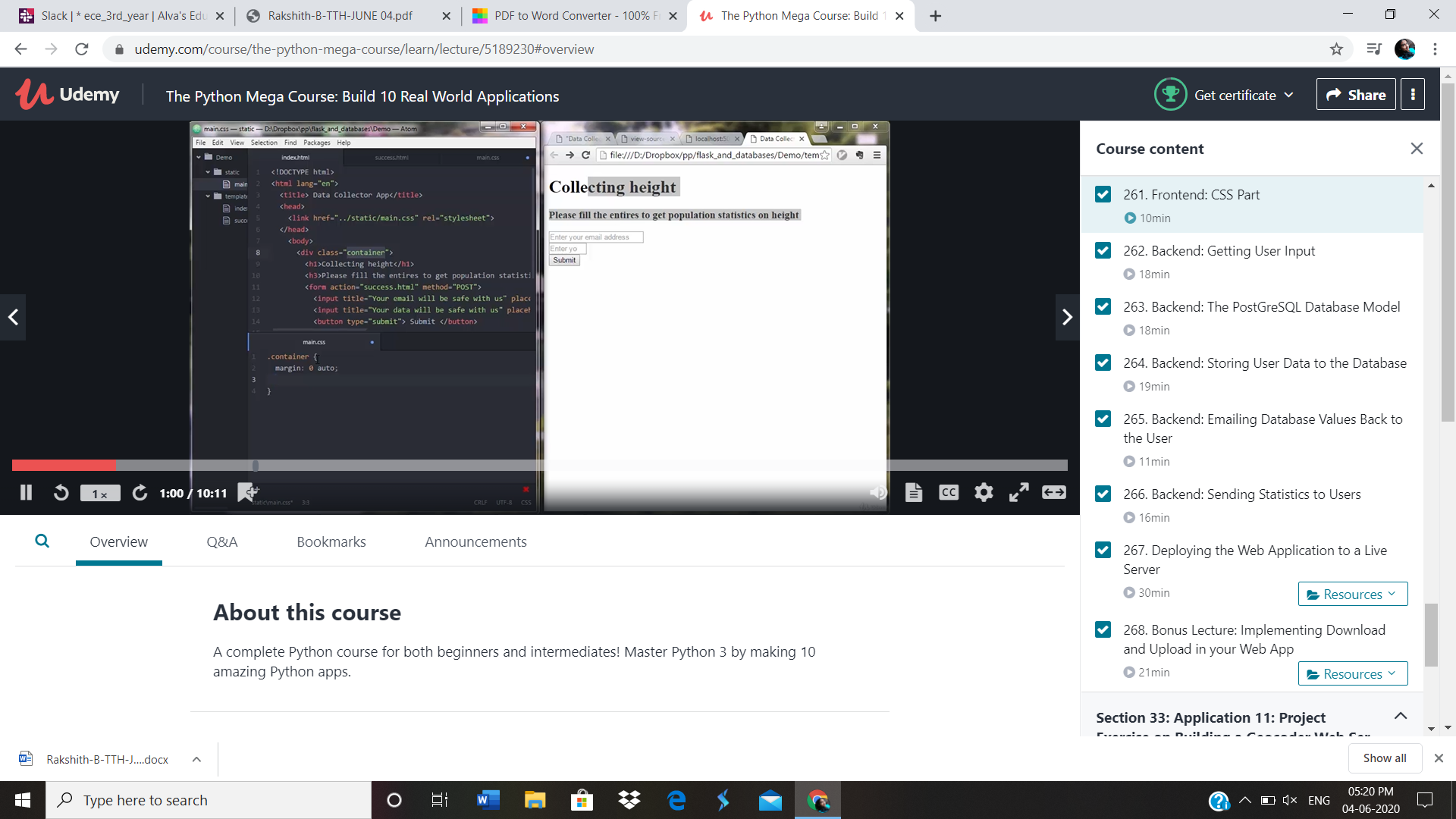
|  |
| --- |
| **FORENOON SESSION DETAILS** |
| **Image of session** |
| **Report –Hardware Modeling Using Verilog Objective of Hardware Modeling Using Verilog**   * **Learn about the Verilog hardware description language.** * **Understand the difference between behavioral and structural design styles.** * **Learn to write test benches and analyze simulation results.** * **Learn to model combinational and sequential circuits,** * **Distinguish between good and bad coding practices.** * **Case studies with some complex designs.**   **VLSI Design Process**   * **Design complexity increasing rapidly**   + **Increased size and complexity**   + **Fabrication technology improving**   + **CAD tools are essential**   + **Conflicting requirements like area, speed, and energy consumption** |

|  |
| --- |
| * **The present trend**   + **Standardize the design flow**   + **Emphasis on low-power design, and increased performance Moore’s Law** * **Exponential growth** * **Design complexity increases rapidly** * **Automated tools are essential** * **Must follow well defined design flow**   **Standardized design procedure**   * **Starting from the design idea down to the actual implementation.**   **Encompasses many steps:**   * **Specification Synthesis** * **Simulation** * **Layout Testability analysis** * **and many more**   **Need to use Computer Aided Design (CAD) tools.**   * **Hardware Description Language (HDL)** * **Based on HDL provide formats for representing the outputs of various design steps** * **A CAD tool transforms its HDL input into a HDL output that contains more detailed information about the hardware.**   + **Behavioral level to register transfer level**   + **Register transfer level to gate level**   + **Gate level to transistor level**   + **Transistor to the layout level Two Competing HDL’s** * **Verilog** * **VHDL**   **Behavioral design**   * **Specify the functionality of the design in terms of its behavior.** * **Various ways of specifying:**   + **Boolean expression or truth table.**   + **Finite-state machine behavior (e.g. state transition diagram or table).**   + **In the form of a high-level algorithm.** * **Needs to be synthesized into more detailed specifications for hardware realization, Data path design** * **Generate a netlist of register transfer level components, like registers, adders, multipliers,**   **multiplexers, decoders, etc.**   * **A netlist is a directed graph, where the vertices indicate components, and the edges**   **indicate interconnections.** |

Name:Bhoomika Hebbar

Course: Python Udemy course USN:4AL17EC010

**Topic: Data Collector Web App Semester & Section:6th A**



app.py

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:postgr es123@localhost/height\_collector'

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"] print(email, height)

if db.session.query(Data).filter(Data.email\_ == email).count()== 0:

data=Data(email,height) db.session.add(data) db.session.commit()

average\_height=db.session.query(func.avg(Data.height\_)).scalar() average\_height=round(average\_height, 1)

count = db.session.query(Data.height\_).count() send\_email(email, height, average\_height, count) print(average\_height)

return render\_template("success.html")

return render\_template('index.html', text="Seems like we got something from that email once!")

if name

== '

main ':