SQUARDO ONLINE BANKING

By: Raj Chhatbar, Siddarth Krishnan, Tavin Chok, Haojie Pan

June 27, 2019

Software Engineering 362

James Choi

TABLE OF CONTENTS

Requirements Analysis (RA)

Chapter 1. Software Requirements (SR)

1. Introduction

a. Purpose

Most of the banking nowadays is done through the internet, as people don't have time to go to the bank and finish their work physically. The Idea is to build an online banking system which will mimic the actual banking system by well-known banks.

Class Name: Software Engineering
***Semester*:** Summer-2019
Group Name: *SquardO*
Members:

Raj Chhatbar
Haojie Pan
Siddarth Krishnan
Tavin Chok

Contribution by Members:

Raj Chhatbar(Implementation, testing, Use-case diagrams, user manual, requirement analysis,Use-case descriptions, Function requirements)
Haojie Pan (Functions, requirements, testing)
Tavin Chok(Implementation, Class Diagrams, use case descriptions, testing)
Siddarth Krishnan(DFD, PSD, Sequence Diagram, and Flow Diagram for code)

b. Scope of the Problem

- **Bank of SquardO**
- Allow the user to sign-up then login and change password. After the user logs in, he or she can deposit, withdraw, transfer money, make an appointment. Plus can reset the password if he/she forgets it by answering the security question. In this app, the user will not be able to pay their bills, and the account is not linked with the debit card.
- The business requirements is to satisfy customers banking requirements at any time and place. Which gave the motivation for building a web based banking system.

- **Business Objectives**
  - Create the website to use bank features
  - Increase the customer base based on new technology
- **Customer or Market Needs**
  - Customers need a platform where they can use banking features 24*7 without wasting their time in a long queue at banks.
- **Business Risks**
  - Provide high level security for each operation. And keeping track of them in real time.

- **<u>Non functional requirements</u>**
- **Performance requirements**
  - As it is a web based system we required a server and depending on server the response time will be less than a second.
- **Platform constraints**
  - We have selected flask framework which can be scalable. However as of now we have selected SQLite database which comes integrated with flask.
- **Accuracy and Precision**
  - The accuracy and precision of the flask are pretty good. It is reliable as its open source there is a good amount of people working on to make it perfect and improve its functionality
- **Reliability**
  - To protect from potential failure we have used flask library called WTForm validation which has many pre verified modules.
- **Security**
  - Security wise the important field such as password and security question are hashed using "sha256" which makes it secure enough in our database.
- **Usability**
  - As we are also using javascript, CSS, and Bootstrap. Due to which system UI is interactive, appealing as well as easy to use because of Nav bar and Dashboard.

C. Definitions, Acronyms, or Abbreviations

- **Online Banking:** Online banking, also known as internet banking, is an electronic payment system that enables customers of a bank or other financial institution to conduct a range of financial transactions through the financial institution's website. The online banking system will typically connect to or be part of the core banking system operated by a bank and is in contrast to branch banking, which was the traditional way customers accessed banking services.
- The user can use it by signing up, where they enter their information in order to get started with system. Once they are logged-in then they can access various features which

are “**Deposit**” where a person can add whatever amount he/she wants to add into their account. “**Withdraw**” is where a person can withdraw whatever amount he/she wants to takeout from their account. “**Transfer**” is where a person can transfer any specified amount from his/her bank account to another person's bank account in that bank. “**Make an appointment**” is where a user can set a date when and where he/she wants to meet bank representative. At last if user forgot the login password we have “**reset password**” feature where he/she can answer their security questions and can change the password.

- **Non functional requirements definitions**
- **Performance requirements**
  - Requirements about resources required, response time, transaction rates, throughput, benchmark specifications or anything else having to do with performance.
- **Platform constraints**
  - Target platform information such as scalability or usability.
- **Accuracy and Precision**
  - Requirements about the accuracy and precision of the data. Accuracy meaning correct data and precision meaning reliable data.
- **Reliability**
  - Requirements about how often the software fails. The measurement is often expressed in MTBF (mean time between failures). It specifies the consequences of software failure, how to protect from failure, a strategy for error detection, and a strategy for correction. Reliability is not availability, that is a different kind of requirement.
- **Security**
  - One or more requirements about protection of the system and its data. The measurement can be expressed in a variety of ways (effort, skill level, time, ...) to break into the system.
- **Usability**
  - Requirements about how difficult it will be to learn and operate the system. The requirements are often expressed in learning time or similar metrics.

D. References

Online banking at Wikipedia. *https://en.wikipedia.org/wiki/Online_banking*

Srs for banking system by Jaydev Kishnani at slideshare.

https://www.slideshare.net/Jaydev_D_Kishnani/srs-for-banking-system

Non-functional Requirements by cal poly pomona.
http://users.csc.calpoly.edu/~jdalbey/SWE/QA/nonfunctional.html

E. Overview

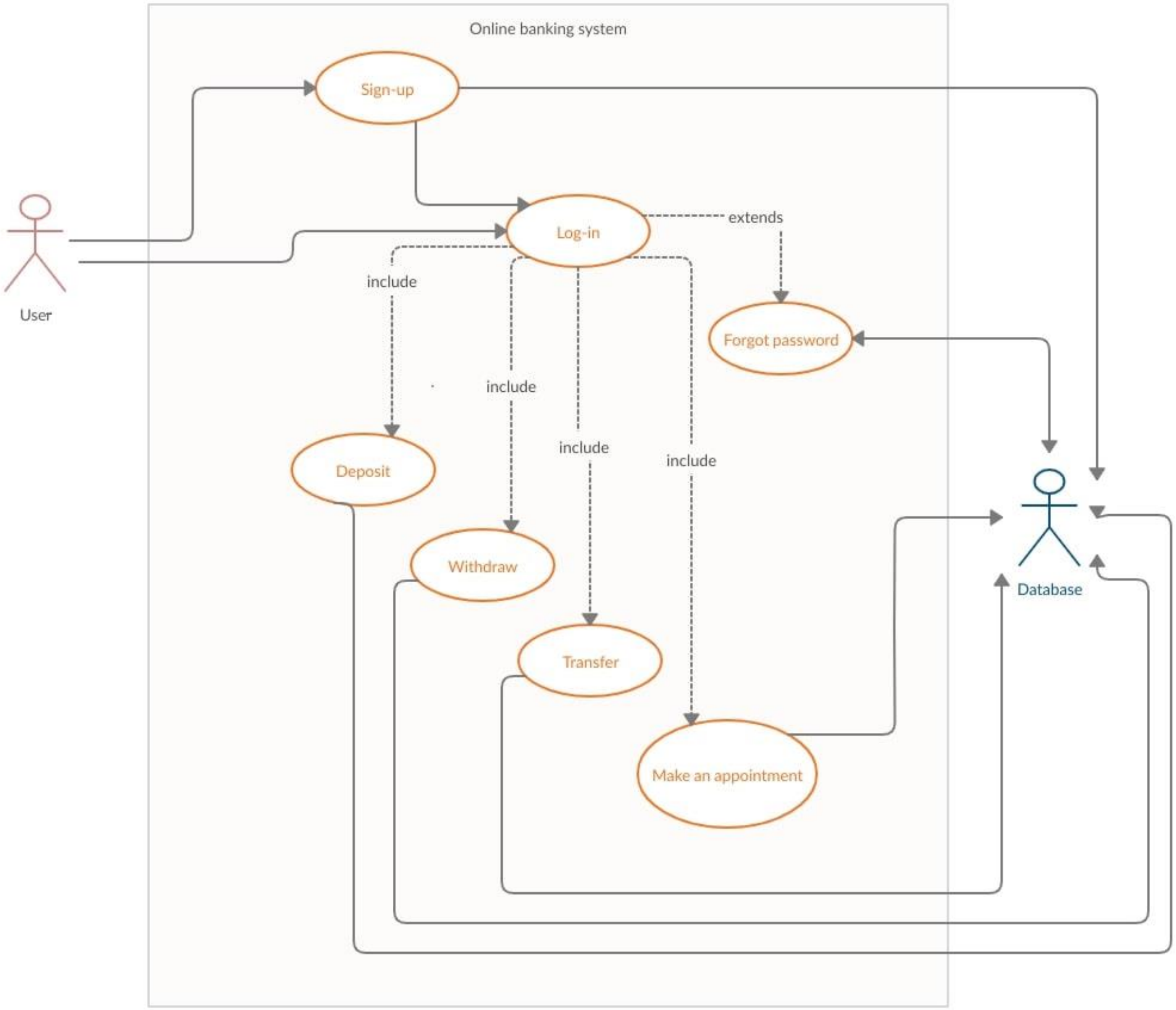
- Form the above system requirements analysis, an overview is. We are designing an online banking system which can handle millions of users at the same time as we are using a framework called flask. Plus the response time will be less than a second, which makes this system very good in terms of performance. Along with that, it is reliable, secure, and easy to use(useable).

1.2 Product Features

FE1: Sign-up
FE2: Login
FE3: Forgot password
FE4: Deposit
FE5: Withdraw
FE6: Transfer
FE7: make an appointment with bank manager

1.3 Use-Case Modeling

a)Use-case diagrams



b)Use-case descriptions

UC-1 Sign-up

Description

The primary function of sign-up is to provide a registration form that a user can fill and get registered as a valid user to use our system.

Primary Actor

The user(customer)

Pre-conditions

Must pass all the validation such as Date,Email, Strong password, length of input etc. for the sign-up form.

UC-2 Login

Description

If a user wants to access their online bank, then they will input their username and password which will grant them access to their specific page where they can use other banking features

Primary Actor

The user(customer)

Pre-conditions

User must have an account otherwise they can’t log-in.

UC-3 Forgot password

Description

If a user forgot his/her password, then they can reset their password by clicking on forgot password link on log-in, which will redirect them to page where they must enter their username, security question answer and then the new password.

Primary Actor

The user(customer)

Pre-conditions

User must have an account otherwise they can’t reset password.

UC-4 Deposit

Description

If a user wants to deposit some money into their account, they can click on “Deposit” where they can enter the total amount they wish to deposit and then by clicking “Deposit” that much amount will be deposited.

Primary Actor

The user(customer)

Pre-conditions

Here user must log-in first to access this feature.

UC-5 Withdraw

Description

If a user wants to withdraw the money from the account, they can click “Withdraw” then specify the amount they wish to withdraw and then click withdraw. Here that much amount will be deducted from his/her account.

Primary Actor

The user(customer)

Pre-conditions

Here user must log-in first to access this feature and they must have a sufficient amount in their account.

UC-6 Transfer Money

Description

If a user wants to transfer the amount to another user within that bank then he/she can click on “Transfer,” and then they must enter the amount plus the account number of the recipient. By clicking transfer, the amount will be transferred from his account to another.

Primary Actor

The user(customer)

Pre-conditions

Here a user must log-in first to access this feature, and they must have a sufficient amount in their account along with that the recipient account number must be valid; otherwise, the transfer will not happen.

UC-7 Make an appointment

Description

If a user wants to meet an adviser from the bank they can make an appointment by clicking on “Make an appointment” where they will enter their details and can schedule an appointment.

Primary Actor

The user(customer)

Pre-conditions

Here a user must log-in first to access this feature.

Functional Requirements

FR1: Sign-Up

a)Functionality

Before start using another feature, the user must register. This can be done using the sign-up function. Here the user has to put his/her details and then click “Sign-up”. Once user click sign-up the whole user data will be stored in SQLite where id(primary key) is his/her account number.

b..) **Realizes :** Sign-up

c.)

Input: username, email, Name, gender, address, city, state, zip code, Date of Birth, phone number, Password, security question, and, Answers to security questions

Output: Thank you for signing up,username

d.)Processing

1) User will go to website
2) Click on sign-up
3) Enter his/her personal information(username, email, Name, gender, address, city, state, zip code, Date of Birth, phone number, Password, security question, and, Answers to security questions)
4) Tell user that sign-up was successful

e) Error Handling

- Every input is required to fill in.
- Each cell takes specific number of values (ex. 4 to 30 for username)
- Check whether email have @ symbol
- Check password have word and numbers
- Check username is unique.

FR2: Login

a)Functionality

It allows the user to access his/her funds and other features. It authenticates whether the user is authentic or not based on username and password as input.

b) Realizes : Log-in

c)

Inputs: username, password

Outputs: Dashboard for user**.**

d)Processing

1. User will go to website
2. Click on Log-in
3. Enter his/her username and password
4. Then user will be redirected to dashboard

e) Error Handling

- Every input is required to fill in.
- Each cell takes specific number of values (ex. 4 to 30 for username)
- It will validate whether that user exist or not
- Plus it will check whether password is right or wrong.

FR3: Forgot Password

a)Functionality

Whenever a user forgets his/her password then they can reset the password based on the security question. Then by providing the right answer to the question they can set a new password.

b) Realizes : Forgot password

c)

Inputs: username, security question answer, New password

Outputs: Acknowledgment that new password is set.

d)Processing

1. Enter username
2. Security questions will be asked.
3. If security question is wrongly answered one again same page will appear
4. Then user can set a new password.

e) Error Handling

- Every input is required to fill in.
- Each cell takes specific number of values (ex. 4 to 30 for username)
- It will validate whether that user exist or not
- Match security question answer with actual answer

FR4: Deposit

a)Functionality

If a user wants to add funds in their account they can use deposit function. And money will be added into users account.

b) Realizes : Deposit

c)

Inputs: Amount in dollars($)

Outputs: Acknowledgment for new deposit

d)Processing

1. From dashboard user should click “Deposit”
2. Then specify amount in dollars($)
3. Click on deposit

e) Error Handling

- Every input is required to fill in.
- Input for dollar amount can’t be negative or 0

FR5: Withdraw

a)Functionality

If a user wants to Withdraw the amount they can do it by using withdraw function. They have to specify the amount which they want to withdraw and then they can withdraw it.

b) Realizes : Withdraw

c)
Inputs: Amount in dollars($)
Outputs: Acknowledgment for new withdrawal

d)Processing

1. From dashboard user should click “Withdraw”
2. Then specify amount in dollars($)
3. Click on withdraw

e) Error Handling

- Every input is required to fill in.
- Input for dollar amount can’t be negative or 0.
- Input amount must be less than or equal to current balance.

FR6: Transfer Money

a)Functionality

If a user wants to transfer an amount from his account to another account within that bank, then he/she can use transfer function where the specified amount will be sent to another user account if a user exists with that account number.

b) **Realizes :**Transfer

c)
Inputs: Amount in dollars($), Recipient account number
Outputs: Acknowledgment for new Transfer

d) **Processing**

1. Click on transfer form dashboard.
2. Enter the amount in dollars you want to transfer.
3. Enter the recipient account number
4. Click on transfer

e) Error Handling

- Every input is required to fill in.
- Input for dollar amount can’t be negative or 0.
- Input amount must be less than or equal to current balance.
- Recipient account number must be valid.

FR7: Make an appointment

a)Functionality

If a user wants to meet an executive from a bank to discuss anything (ex. Credit card, loan, etc.), then he/she can schedule an appointment form the online system. Where the user will specify at what time and place he/she wants to be there.

b) **Realizes :**Make an appointment

c)
Inputs: date, time, location, about what?
Outputs: Show upcoming appointment.

d) Processing

1. **From dashboard click “Make an appointment”**
2. **Enter the required input**
3. **Press schedule it**

e) Error Handling

- Every input is required to fill in.
- Check for date and time value

External Interfaces

User Interfaces

The user interaction will be based on select feature by clicking over them and inputing values using keyboard.
Some of the functions are dashboard, Log-in etc.

Hardware Interfaces

Server should be set up inorder to provide the service.
But at client side any computer will work

Software Interfaces

Its a web based software so can run on any platform(Windows, Linux, etc.) including mobile phones.

Communications Interfaces

The whole communication is done by web framework flask where we are using mainly POST method to communicate. And it is been handled by web browser.

Performance Requirements

The performance of our product is based on flask and SQLite. As there are many pre built libraries we can scale the system to millions of user provided we have enough computing power. Along with that SQLite can work but is limited. So later database can be shifted to mysql or oracle sql.

a) Reliability Because our app will work under all types of computers the reliability is high.
b) Response Time less than one second.

1.7 Other Requirements

a) Time-The time consumption of our app is around 3 weeks.
b) Cost - As we are using open source software such as flask and SQLite it's free of cost
c) Site Adaptation-N/A
d) Security- The flask server comes with many prebuild security modules such as hashing in "sha256", ddos attack etc.
e) any other-N/A

Chapter 2. Specification/Analysis Modeling

2.1 Introduction

a) **Purpose**

The purpose of this chapter is to delve into the requirements further and interpret what they mean.

b) **Definitions, Acronyms, or Abbreviations**

DFD(data flow diagram**)**

In Software engineering DFD(data flow diagram) can be drawn to represent the system of different levels of abstraction. Higher level DFDs are partitioned into low levels-hacking more information and functional elements. Levels in DFD are numbered 0, 1, 2 or beyond.

Class diagrams
Class diagrams are the main building blocks of every object oriented methods. The class diagram can be used to show the classes, relationships, interface, association, and collaboration.

c) References

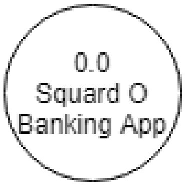
Most of the reference was based on powerpoints from lectures.

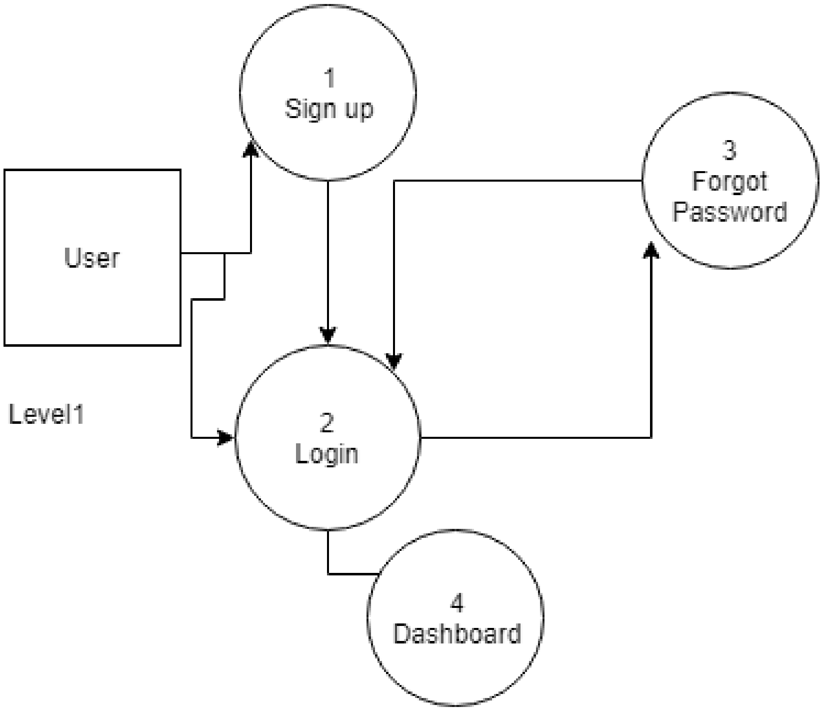
Data flow diagram by geeksforgeeks.com
https://www.geeksforgeeks.org/levels-in-data-flow-diagrams-dfd/

Class Diagrams by geeksforgeeks.com
https://www.geeksforgeeks.org/unified-modeling-language-uml-class-diagrams/

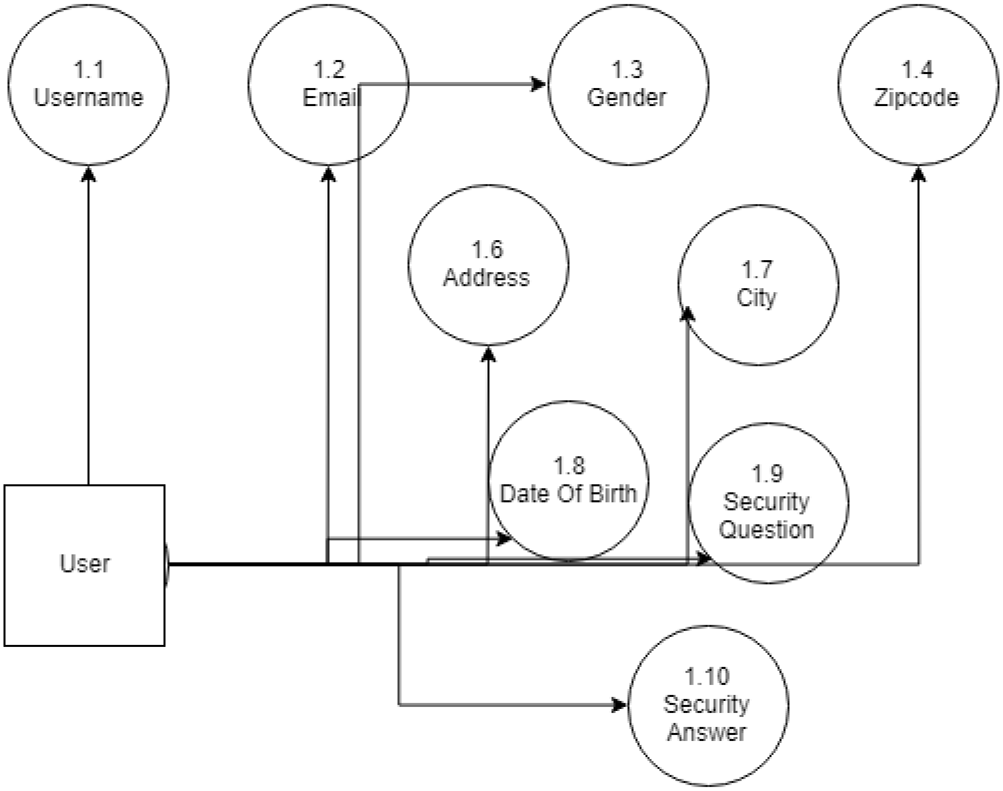
2.2 Data Flow Diagrams

Level 0

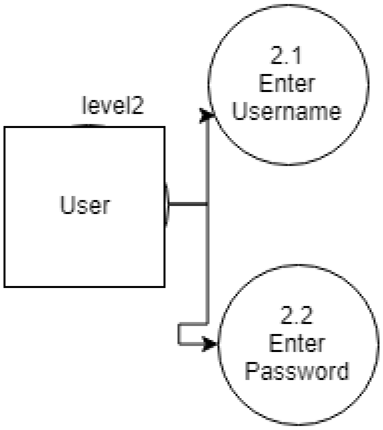




Level2

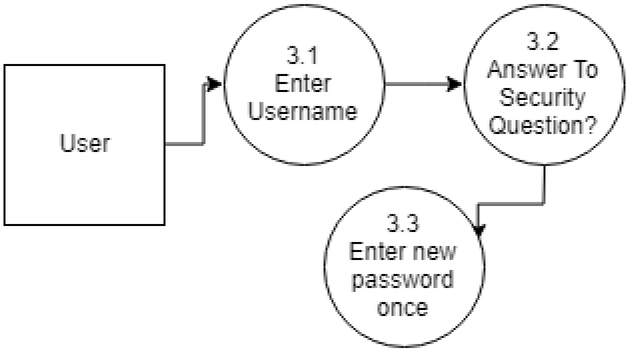


1.1
Username
1.2
Email
1.3
Gender
1.4
Zipcode
1.6
Address
1.7
City
1.8
Date Of Birth
1.9
Security
Question
User
1.10
Security
Answer

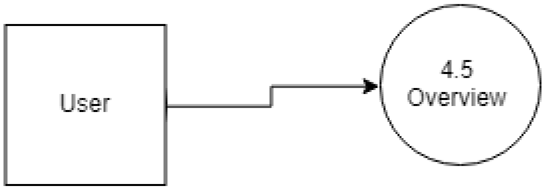


2.1
Enter
Username
level2
User
2.2
Enter
Password

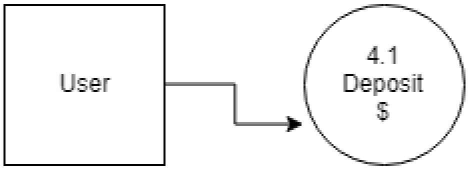
level2



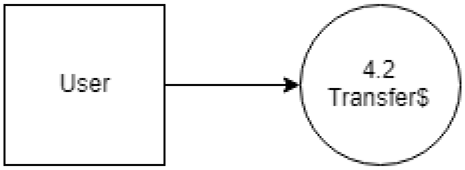
level2



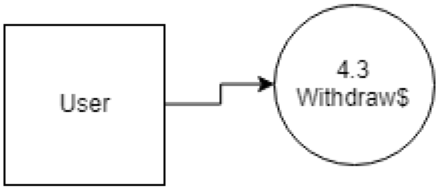
level2



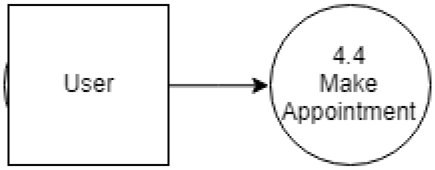
level2



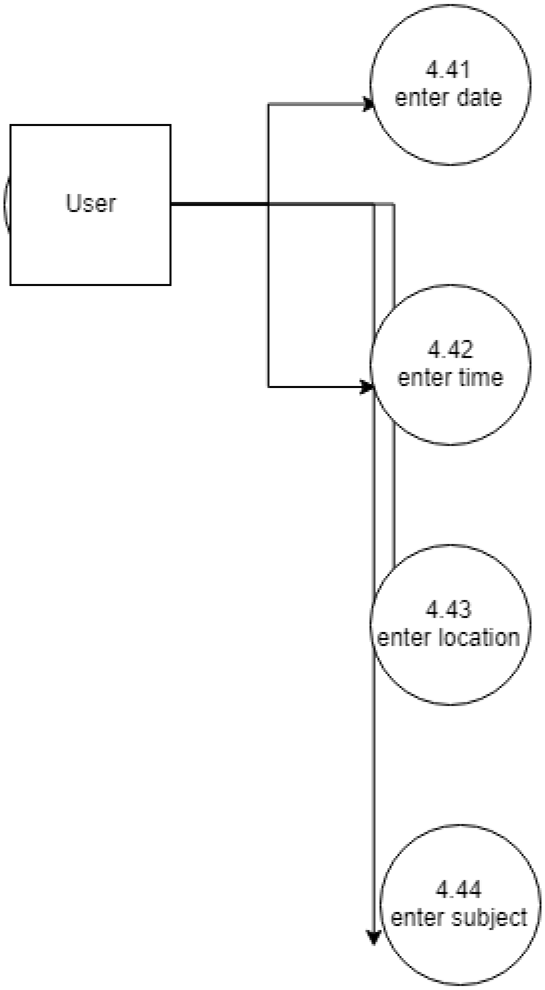
level2



level2

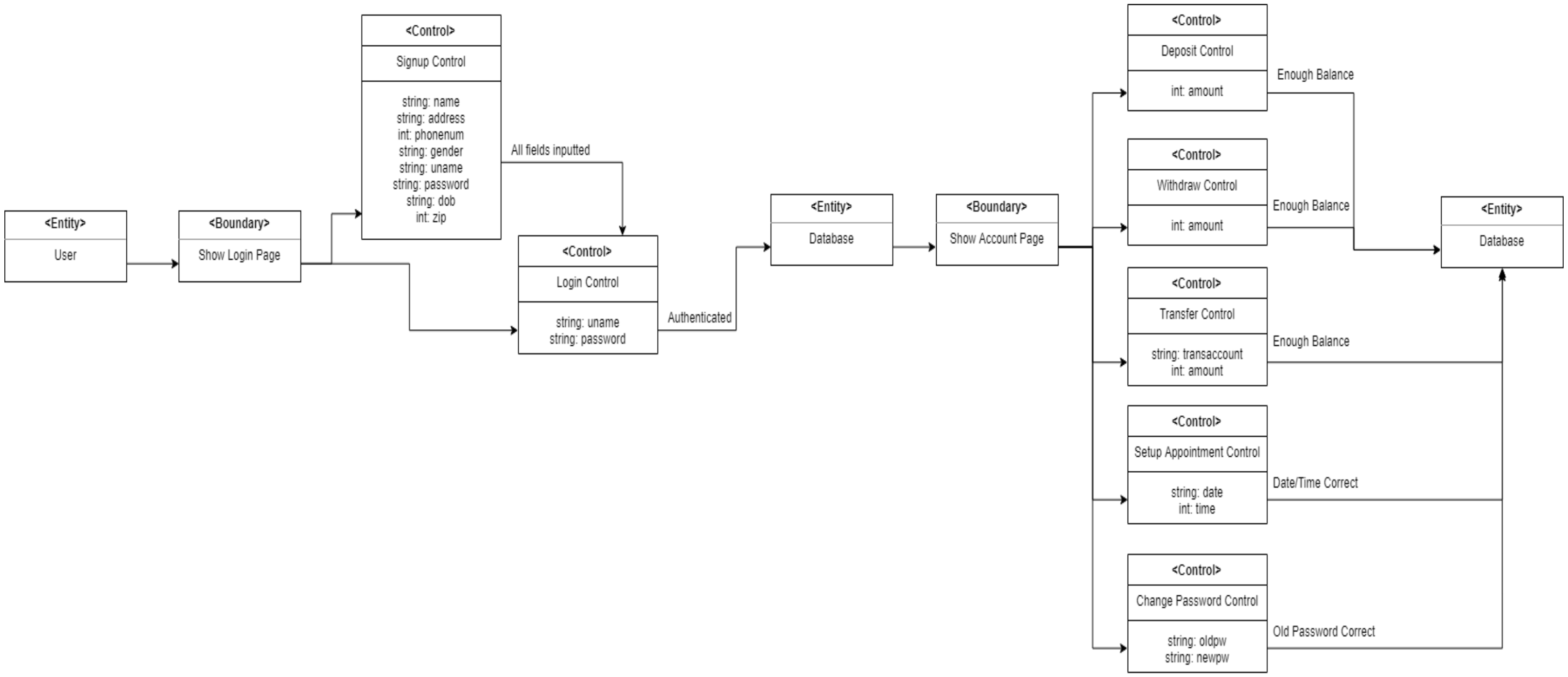


level3

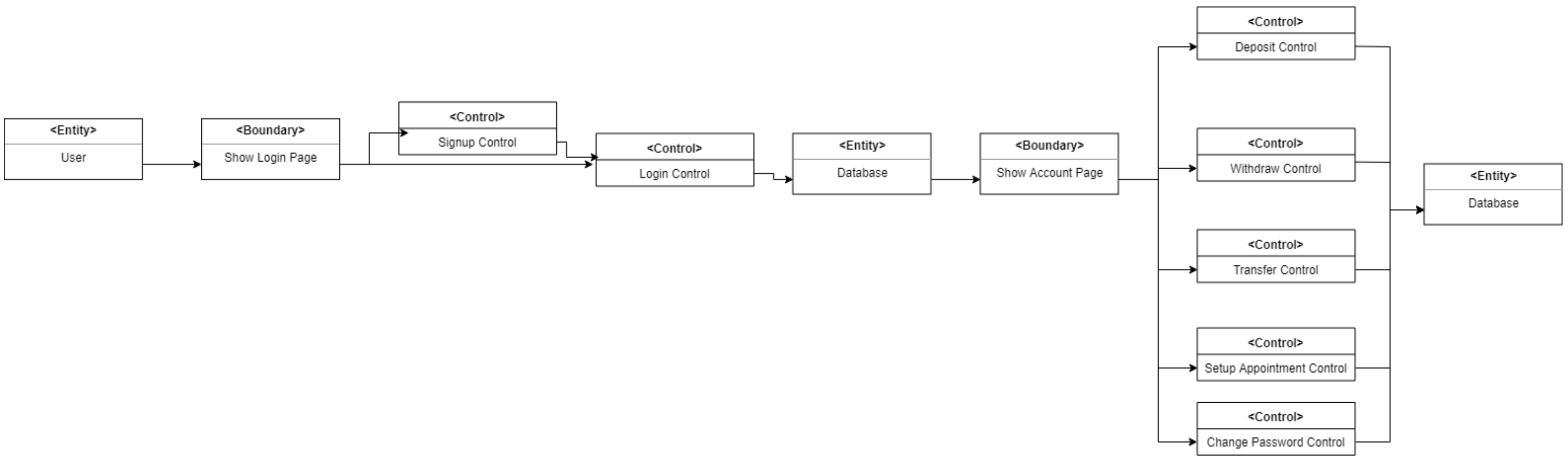


2.3 Class Modeling

a) Initial Class Diagram



b) Simplified Class Diagram



Chapter 3. Design Modeling

3.1 Introduction

a) Purpose- The purpose of this chapter is to translate DFD into PSD ,create a sequence diagram and detailed class diagrams.

b) Definitions, Acronyms or Abbreviations-

DFD- Data Flow Diagram.
PSD-Program Structure Diagram.
UN=username
PW=password
SQ=security question
SQA= security question answer
DOB=Date of birth

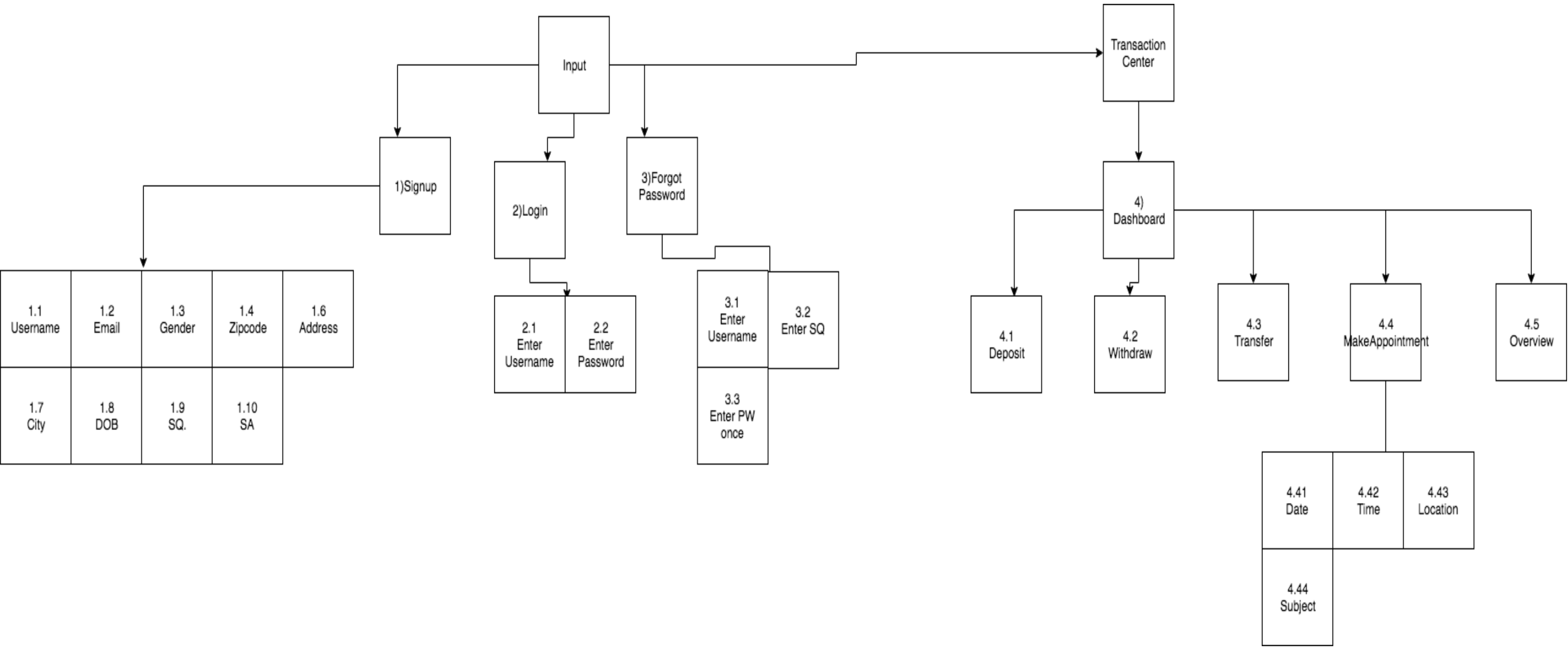
c) References-

Most of the reference was based on powerpoints from lectures.

d) Overview-N/A.

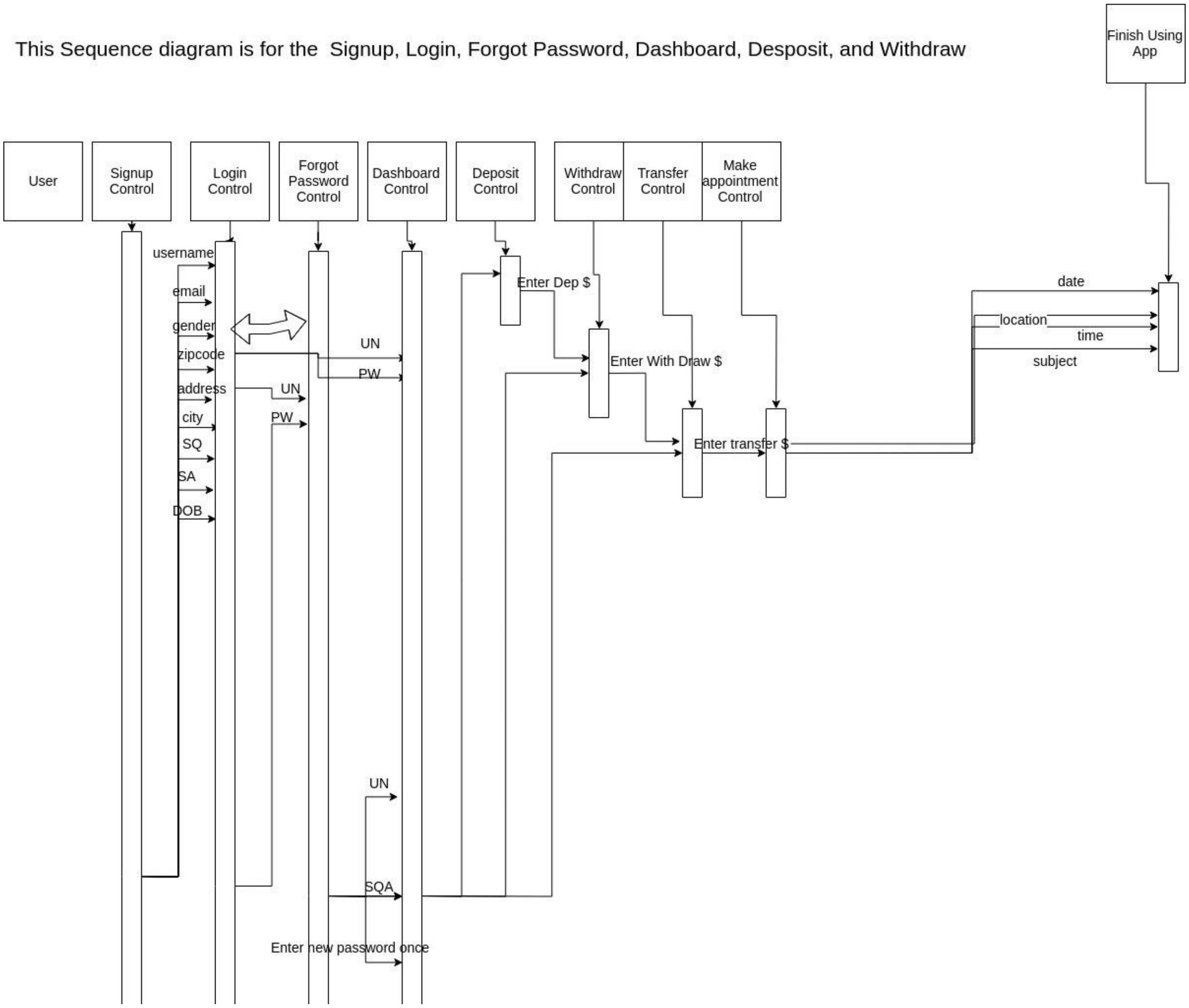
3.2. Functional Modeling

Program Structure Diagram



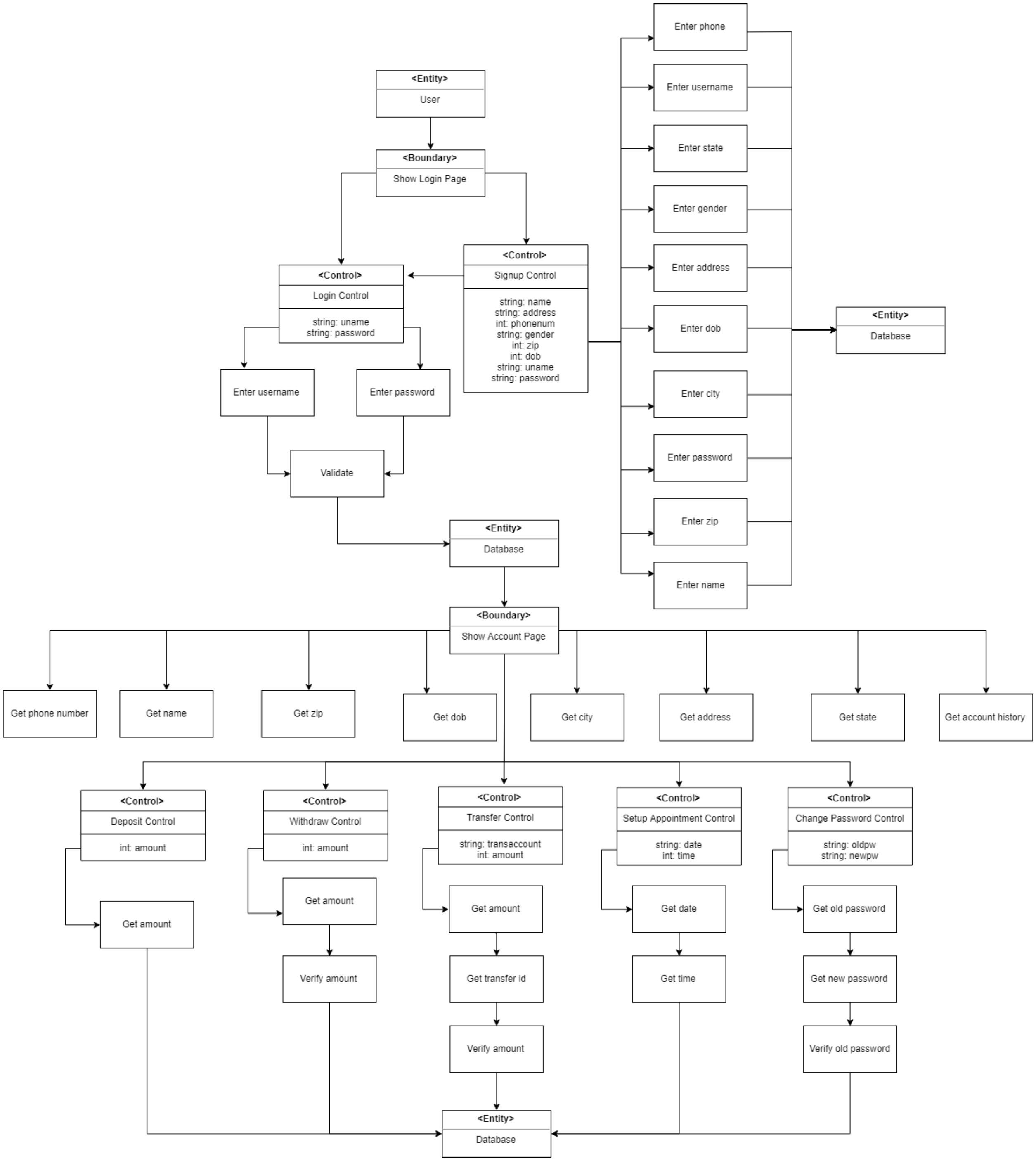
3.3 OO modeling

a) Sequence Diagrams



This Sequence diagram is for the Signup, Login, Forgot Password, Dashboard, Desposit, and Withdraw

b) Detailed Class Diagrams



Chapter 4. Implementation

4.1. Introduction

a) Purpose

The purpose of this section is to show our source code for online banking system

The implementation was started around 15 June 2019 by Raj Chhatbar and Tavin Chok and finished on 25 June 2019.

b) Definitions, Acronyms or Abbreviations

None

c) References

Python and Flask Bootcamp by Jose Portilla
https://www.udemy.com/course/python-and-flask-bootcamp-create-websites-using-flask/

Python Flask From Scratch User Registration by traversy media
https://www.youtube.com/watch?v=addnlzdSQs4

Plus many different questions from stackoverflow.com
https://stackoverflow.com/

4.2. Implementation Environment

The whole code was written in python. And a specific module named flask which is a web framework was used. And for database we used SQLite as it is integrated with flask. The virtual environment was setup in Linux to take care of different requirements of each package. Whole development was done in Ubuntu operating system.
To set up virtual environment in Linux it can be done using

```
Sudo apt-pip install virtualenv
```

Then create environment using

```
virtualenv -p  myproject
```

Then to activate virtual environment

```
Source bin/activate
```

And then install all requirements using requirement.txt

```
Pip -r install requirments.txt
```

The reset of visual was implemented using HTML, CSS, Bootstrap and Javascript.

Chapter 5. Testing

5.1 Introduction

a) Purpose – The purpose of this chapter is to test 1 module/method in our program.

b) Definitions, Acronyms or Abbreviations- N/A.

c) References – N/A.

d) Overview – We are going to show the signup function in our program and the flow graph of our code.

5.2 Module Testing

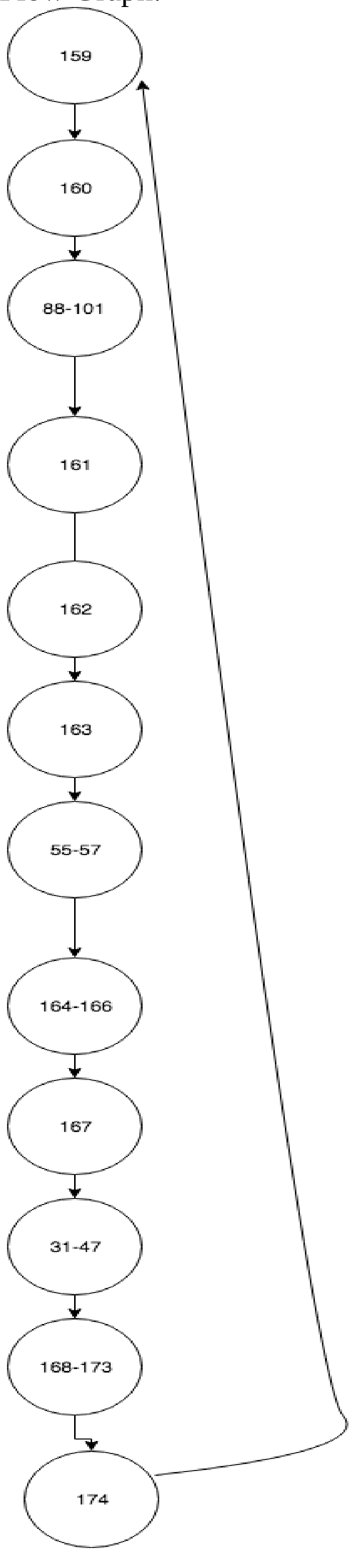
a)

```
class User(UserMixin, db.Model):

    id = db.Column(db.Integer ,primary_key=True, autoincrement = True)
    username = db.Column(db.String(15), unique=True)
    name=db.Column(db.String(15))
    gender=db.Column(db.String(7))
    address=db.Column(db.String(130))
    city=db.Column(db.String(100))
    state=db.Column(db.String(80))
    zipcode=db.Column(db.String(80))
    date_of_birth=db.Column(db.Date)
    phone_number=db.Column(db.String(15))
    security_question=db.Column(db.String(150))
    security_question_answer=db.Column(db.String(150))

    email = db.Column(db.String(50), unique=True)
    password = db.Column(db.String(80))

class Account(db.Model):
    account_id=db.Column(db.Integer , db.ForeignKey('user.id'), primary_key=True,autoincrement = True)
    balance=db.Column(db.Float)

class RegisterForm(FlaskForm):
    email = StringField('email', validators=[InputRequired(), Email(message='Invalid email'), Length(max=50)])
    username = StringField('username', validators=[InputRequired(), Length(min=4, max=15)])
    name = StringField('name', validators=[InputRequired(), Length(min=2, max=60)])
    gender = RadioField('gender', choices = [('male','Male'),('female','Female'),('other','Other')], validators=[InputRequired()])
    address=TextAreaField("Address",validators=[InputRequired(), Length(min=4, max=130)])
    city=StringField("city",validators=[InputRequired(), Length(min=2, max=100)])
    state=StringField("state",validators=[InputRequired(), Length(min=2, max=80)])
    zipcode=StringField("zip-code",validators=[InputRequired(), Length(min=2, max=80)])
    date_of_birth=DateField("Date of Birth",validators=[InputRequired()], format='%m/%d/%y')
    phone_number=StringField("Phone No.",validators=[InputRequired(), Length(min=4, max=15)])
    security_question=StringField("Security question",validators=[InputRequired(), Length(min=4, max=150)])
    security_question_answer=StringField("Security question answer",validators=[InputRequired(), Length(min=4, max=150)])

    password = PasswordField('password', validators=[InputRequired(), Length(min=8, max=80)])
```

```
def signup():
    form = RegisterForm()

    if form.validate_on_submit():
        hashed_password = generate_password_hash(form.password.data, method='sha256')
        new_user = User(username=form.username.data,name=form.name.data, email=form.email.data, gender=form.gender.data,address=form.address.data,city=form.city.data,state=form.state.data, zipcode=form.zipcode.data,date_of_birth=form.date_of_birth.data,phone_number=form.phone_number.data,security_question=form.security_question.data,security_question_answer=form.security_question_answer.data ,password=hashed_password)
        new_account=Account(balance=0.0)
        #db.create_all()

        db.session.add(new_user)
        db.session.add(new_account)
        db.session.commit()

        return '<h1>New user has been created!</h1>'
        #return '<h1>' + form.username.data + ' ' + form.email.data + ' ' + form.password.data + '</h1>'

    return render_template('signup.html', form=form)
```

b) Independent Path Testing

- Flow Graph:



- Paths: 1)159,174 2) 159-174
  - Testcase-Enter any empty field.
  - Expected Value-Not let the user proceed to login with their signed-up information.

- Test Results. doesn’t let the user proceed with empty credentials.
- Conclusion: Passed

Chapter 6 User Manual

6.1. Introduction

a) Purpose – The purpose of this chapter is to tell the user what he or she needs to run the program and how to make the program work.

b) Definitions, Acronyms or Abbreviations – N/A.

c) References – N/A.

d) Overview – Shows the user how to correctly use the application.

6.2. Hardware Configuration

The user needs a working keyboard, mouse, and computer.

6.3. System Parameters

The system can either be a high-end computer or low-end computer.

6.4. Operation Procedure

1) Place the requirements.txt and app.py into desired destination
2) run the following command on the terminal - sudo apt install pip
3) Navigate to “desired destination”
4) pip install -r requirements.txt.
5) run python app.py
6) Go to http://127.0.0.1:5000/

6.5. Demonstration

Show your working system.

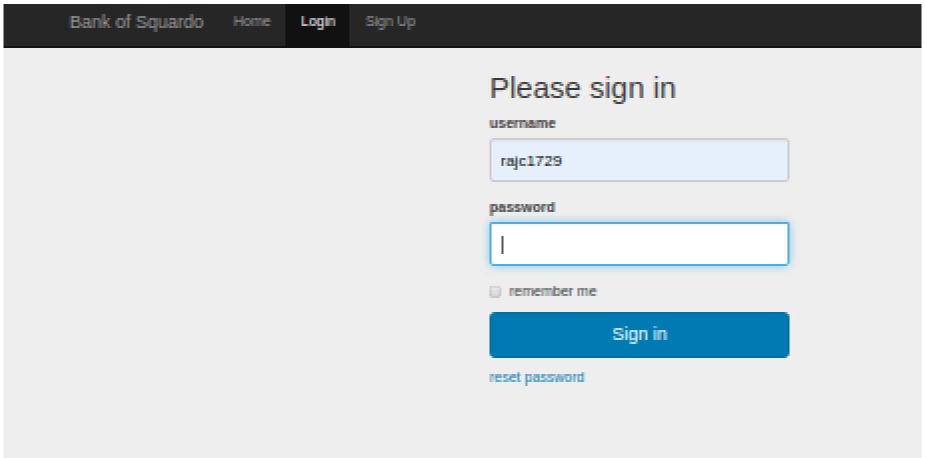
<u>Select any one requirement (feature)</u> and execute it from start to the end.

Attach screen dumps for that requirement.

Here we are demonstrating our forgot password feature.

<u>Feature</u> : Forgot password

a) At first got to login page

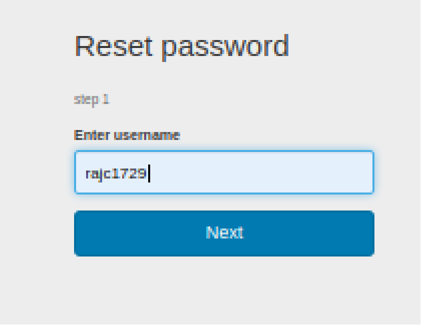


Please sign in

username: rajc1729

password:

b) Now press reset password as we don’t know our current password

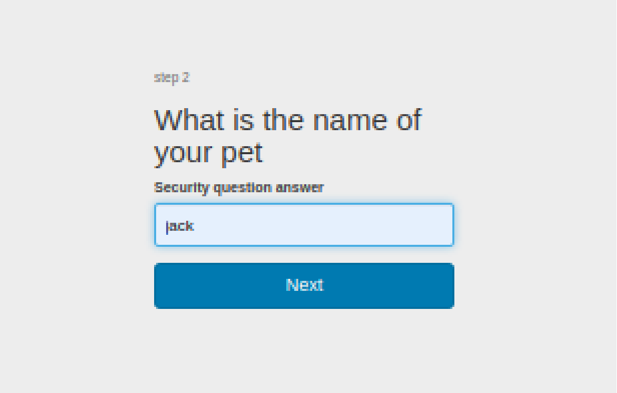


Reset password

step 1

Enter username: rajc1729

c) Now click new to step number 2 where we have to answer our security question

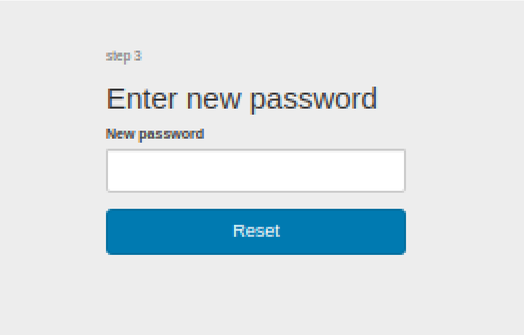


step 2

What is the name of your pet

Security question answer: jack

d) Now after answering security question answer if it matches the original answer than new password prompt will appear


step 3

Enter new password

New password

e) Once the new password is set now a user can login with new password.

Appendix A - Group Reports

Group Meeting Report 1

Reporter: Raj Chhatbar
Date: Starting Time: Ending Time: 2:00 PM to 3:30 PM
Participants: Raj, Haojie, Tavin, Siddarth
Missing:

Discussion Topic: Online Banking

Discussion Content (use extra sheets if necessary):
We completed sections 1(Introduction) and 2(Product Features) of the Requirements Analysis Sheet.

Reviews Conducted (use extra sheets if needed):
N/A.
Problems identified/Proposed Solutions (use extra sheets if needed):
The problems we identified were too many features on the app. So we revised our solution, and cut down the number of features.

Any other comments:

Group Meeting Report 2

Reporter: Raj Chhatbar
Date: Starting Time: Ending Time: 2:00 PM to 3:30 PM
Participants: Raj, Haojie, Tavin, Siddarth

Missing:

Discussion Topic: Online Banking

Discussion Content (use extra sheets if necessary):

We completed sections 1(Introduction) and 2(Product Features) of the Requirements Analysis Sheet. The most of the work we did was building use case diagrams and function requirement.

Reviews Conducted (use extra sheets if needed):
The most of the review where for features and their reuirements.

Problems identified/Proposed Solutions (use extra sheets if needed):
If balance is 0, then notify the user with the message and have the user redo the withdraw process.
If the username is wrong, then notify the user with a message and have the user redo the login process.
If the password is wrong, then notify the user with a message and have the user redo the login process.

Any other comments:

Group Meeting Report 3

Reporter: Raj Chhatbar
Date: Starting Time: Ending Time: 2:00 PM to 3:30 PM
Participants: Raj, Haojie, Tavin
Missing:

Discussion Topic: Online Banking, UML diagram, DFD diagram

Discussion Content (use extra sheets if necessary):
The most of the work we did was building use case diagrams and function requirement. Plus now we are focusing on features as well as their implementation.

Reviews Conducted (use extra sheets if needed):
The most of the review where for features and their requirements.

Problems identified/Proposed Solutions (use extra sheets if needed):
If balance is 0, then notify the user with the message and have the user redo the withdraw process.
If the username is wrong, then notify the user with a message and have the user redo the login process.
If the password is wrong, then notify the user with a message and have the user redo the login process.

Any other comments:

Group Meeting Report 4

Reporter: Tavin Chok
Date: Starting Time: Ending Time: 2:00 PM to 3:30 PM
Participants: Raj, Haojie, Tavin, siddhart
Missing:

Discussion Topic: Online Banking, UML diagram, DFD diagram

Discussion Content (use extra sheets if necessary):
We worked on the data flow diagram of Phase 2.

Reviews Conducted (use extra sheets if needed):

Problems identified/Proposed Solutions (use extra sheets if needed):

Any other comments:

Group Meeting Report 5

Reporter: siddhart
Date: Starting Time: Ending Time: 2:00 PM to 3:30 PM
Participants: Raj, Haojie, Tavin, siddhart
Missing:

Discussion Topic: Online Banking, UML diagram, DFD diagram

Discussion Content (use extra sheets if necessary): N/A

Reviews Conducted (use extra sheets if needed): Reviewed phase 1 and phase 2 documents and filled in the missing blaking.

Problems identified/Proposed Solutions (use extra sheets if needed): We found that we didn't fill out the interfaces portion of the document on phase 2 so we got that done.

Any other comments: Siddarth Krishnan finished the Diagram(DFD) and Evan finished the class diagram.

Group Meeting Report 7

Reporter: siddhart
Date: Starting Time: Ending Time: 2:00 PM to 3:30 PM
Participants: Raj, Haojie, Tavin, siddhart
Missing:

Discussion Topic: Online Banking, UML diagram, DFD diagram

Discussion Content (use extra sheets if necessary): N/A

Reviews Conducted (use extra sheets if needed): We reviewed phase 1 and phase 2 and fixed the mistakes on it based on the teacher’s suggestions. Phase 3 was also completed.

Problems identified/Proposed Solutions (use extra sheets if needed):. We found out that the PSD and DFD weren’t done correctly. We made the necessary changes and fixed it the way the professor wants.

Any other comments:

Appendix B - Source Code

////////////////////////////////Backend of software//////////////////////////////

All data in python are stored using python list(*data structure*) when copied from SQLite.

App.py

```
# these are modules for running program

from flask import Flask, render_template, url_for,session,redirect
from flask_bootstrap import Bootstrap
from flask_wtf import FlaskForm
from wtforms import StringField, PasswordField,DateTimeField
,DateField,TextAreaField,BooleanField,RadioField,IntegerField,FloatField, SubmitField
from wtforms_components import TimeField
from wtforms.validators import InputRequired, Email, Length
from flask_sqlalchemy  import SQLAlchemy
from werkzeug.security import generate_password_hash, check_password_hash
from flask_login import LoginManager, UserMixin, login_user, login_required, logout_user,
current_user
import os
import time
import datetime
from sqlalchemy.ext.declarative import declarative_base
from flask_admin.contrib.sqla import ModelView
from flask_admin import Admin
from flask import g



Base = declarative_base()
```

```
app = Flask(__name__) # created flask app called app
app.config['SECRET_KEY'] = 'Thisissupposedtobesecret!' # secret key for security reasons

base_dir = os.path.abspath(os.path.dirname(__file__)) # get file path from base directory
app.config['SQLALCHEMY_DATABASE_URI'] =
'sqlite:///'+os.path.join(base_dir,'data.sqlite')
app.config['SQLALCHEMY_TRACK_MODIFICATIONS']=False
bootstrap = Bootstrap(app)
db = SQLAlchemy(app)  # create database
login_manager = LoginManager()
login_manager.init_app(app)
login_manager.login_view = 'login'
admin=Admin(app)
# user class for storing sign-up form data in SQLite

class User(UserMixin, db.Model):
# date 15-june-2019     @author RAJ CHHATBAR
    id = db.Column(db.Integer ,primary_key=True, autoincrement = True)
    username = db.Column(db.String(15), unique=True)
    name=db.Column(db.String(15))
    gender=db.Column(db.String(7))
    address=db.Column(db.String(130))
    city=db.Column(db.String(100))
    state=db.Column(db.String(80))
    zipcode=db.Column(db.String(80))
    date_of_birth=db.Column(db.Date)
    phone_number=db.Column(db.String(15))
    security_question=db.Column(db.String(150))
    security_question_answer=db.Column(db.String(150))

    email = db.Column(db.String(50), unique=True)
    password = db.Column(db.String(80))


# Account class for storing account balance in SQLite

class Account(db.Model):
# date 15-june-2019     @author RAJ CHHATBAR
    account_id=db.Column(db.Integer , db.ForeignKey('user.id'),
primary_key=True,autoincrement = True)
    balance=db.Column(db.Float)



# Transaction class for storing each transaction
```

```
class Transaction(db.Model):
# date 16-june-2019     @author RAJ CHHATBAR
    transfer_id= db.Column(db.Integer ,primary_key=True, autoincrement = True)
    account_id=db.Column(db.Integer,db.ForeignKey('account.account_id'))
    amount=db.Column(db.Float)
    balance=db.Column(db.Float)
    time=db.Column(db.DateTime)
    type=db.Column(db.String(30))

#Make_an_appointment class for storing appointments
# date 22-june-2019     @author RAJ CHHATBAR
class Make_an_appointment(db.Model):
    appointment_id=db.Column(db.Integer ,primary_key=True, autoincrement = True)
    appointment_account_id=db.Column(db.Integer,db.ForeignKey('user.id'))
    appointment_date=db.Column(db.Date)
    appointment_time=db.Column(db.Time)
    appointment_location=db.Column(db.String(60))
    about_what=db.Column(db.String(100))



# get current login user
@login_manager.user_loader
def load_user(user_id):
    return User.query.get(int(user_id))

# class LoginForm for user login

class LoginForm(FlaskForm):
# date 15-june-2019     @author TAVIN CHOK
    username = StringField('username', validators=[InputRequired(), Length(min=4, max=15)])
    password = PasswordField('password', validators=[InputRequired(), Length(min=8, max=80)])
    remember = BooleanField('remember me')



# class sign-up form for user signup
# date 15-june-2019     @author RAJ CHHATBAR
class RegisterForm(FlaskForm):
    email = StringField('email', validators=[InputRequired(), Email(message='Invalid email'), Length(max=50)])
    username = StringField('username', validators=[InputRequired(), Length(min=4, max=15)])
    name = StringField('name', validators=[InputRequired(), Length(min=2, max=60)])
    gender = RadioField('gender', choices = [('male','Male'),('female','Female'),('other','Other')], validators=[InputRequired()])
    address=TextAreaField("Address",validators=[InputRequired(), Length(min=4,
max=130)])
    city=StringField("city",validators=[InputRequired(), Length(min=2, max=100)])
    state=StringField("state",validators=[InputRequired(), Length(min=2, max=80)])
    zipcode=StringField("zip-code",validators=[InputRequired(), Length(min=2, max=80)])
    date_of_birth=DateField("Date of Birth",validators=[InputRequired()],
format='%m/%d/%y')
    phone_number=StringField("Phone No.",validators=[InputRequired(), Length(min=4,
max=15)])
    security_question=StringField("Security question",validators=[InputRequired(),
Length(min=4, max=150)])
    security_question_answer=StringField("Security question
answer",validators=[InputRequired(), Length(min=4, max=150)])

    password = PasswordField('password', validators=[InputRequired(), Length(min=8,
max=80)])

# date 16-june-2019     @author TAVIN CHOK
# class deposit form form deposit form input
class DepositForm(FlaskForm):
    amount_deposit = FloatField('Deposit amount', validators=[InputRequired()])
    deposit=SubmitField("Deposit")


# date 16-june-2019     @author RAJ CHHATBAR
# class withdrawalForm form withdraw form input
class WithdrawForm(FlaskForm):
    amount_withdraw = FloatField('Withdraw amount', validators=[InputRequired()])
    withdraw=SubmitField("Withdraw")

# date 17-june-2019     @author TAVIN CHOK
# class TransferForm form form Transfer form input
class TransferForm(FlaskForm):
    amount_transfer = FloatField('Transfer amount', validators=[InputRequired()])
    account_number=IntegerField('Account number', validators=[InputRequired()])
    transfer=SubmitField("Transfer")

# date 22-june-2019     @author RAJ CHHATBAR
# class AppointmentForm form Appointment form input
class AppointmentForm(FlaskForm):
    appointment_date=DateField("Date" ,validators=[InputRequired()], format='%m/%d/%y')
    appointment_time=TimeField("Time" ,validators=[InputRequired()])
    appointment_location=StringField('location', validators=[InputRequired(), Length(min=2,
max=60)])
    about_what=StringField('About what', validators=[InputRequired(), Length(min=2,
max=100)])
```

```
    schedule=SubmitField("Schedule")

# date 23-june-2019     @author RAJ CHHATBAR
# class ResetForm form Reset form input
class ResetForm(FlaskForm):
    reset_username = StringField('Enter username', validators=[InputRequired(),
Length(min=4, max=15)])
    next=SubmitField("Next")

# date 23-june-2019     @author RAJ CHHATBAR
# class SecurityForm form Security form input
class SecurityForm(FlaskForm):
    question_answer=StringField("Security question answer",validators=[InputRequired(),
Length(min=4, max=150)])

# date 23-june-2019     @author RAJ CHHATBAR
# class ResetPasswordForm form Security form input
class ResetPasswordForm(FlaskForm):
    password = PasswordField('New password', validators=[InputRequired(), Length(min=8,
max=80)])




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

# date 15-june-2019     @author TAVIN CHOK
@app.route('/login', methods=['GET', 'POST'])
def login():
    form = LoginForm()      #create login form object

    if form.validate_on_submit():
        user = User.query.filter_by(username=form.username.data).first() # find user in
database
        if user:
            if check_password_hash(user.password, form.password.data):  # check hash
password
                login_user(user, remember=form.remember.data)
                return redirect(url_for('dashboard')) # redirect to dashboard

        return '<h1>Invalid username or password</h1>' # if incorrect display message


    return render_template('login.html', form=form)
```

```
# date 15-june-2019     @author TAVIN CHOK
@app.route('/signup', methods=['GET', 'POST'])
def signup():
    form = RegisterForm()   # create registration form object

    if form.validate_on_submit():
        hashed_password = generate_password_hash(form.password.data, method='sha256') #
generate hash password
        new_user = User(username=form.username.data,name=form.name.data,
email=form.email.data,
gender=form.gender.data,address=form.address.data,city=form.city.data,state=form.state.dat
a,zipcode=form.zipcode.data,date_of_birth=form.date_of_birth.data,phone_number=form.ph
one_number.data,security_question=form.security_question.data,security_question_answer=f
orm.security_question_answer.data ,password=hashed_password) # create new user object
        new_account=Account(balance=0.0)  # set balance to zero



        db.session.add(new_user)  # add user to session
        db.session.add(new_account) # add account details to sessions
        db.session.commit()  # commit changes to database

        return '<h1>New user has been created!</h1>'  #message for new user



    return render_template('signup.html', form=form) # display html file plus get user input

@app.route('/dashboard')  # set url for dashboard
@login_required   # loginrequired to accessing dashboard

# date 15-june-2019     @author RAJ CHHATBAR
def dashboard():
    user = User.query.filter_by(username=current_user.username).first() # get user from
database
    account = Account.query.filter_by(account_id=user.id).first()  #get account form database
    all_transactions= Transaction.query.filter_by(account_id=user.id) #get all transactions
from database
    return render_template('dashboard.html', all_transactions=all_transactions,id=user.id
,user_name=current_user.username,name=user.name,address=user.address,city=user.city,stat
e=user.state,zipcode=user.zipcode,phone=user.phone_number,balance=account.balance )
# here all variable are taking input from form HTML


# date 16-june-2019     @author TAVIN CHOK
@app.route('/deposit', methods=['GET', 'POST'])
@login_required
```

```
def deposit():

    form=DepositForm()   # form class for deposit
    user = User.query.filter_by(username=current_user.username).first() # get user from
database
    account = Account.query.filter_by(account_id=user.id).first() #get account form database
    if form.validate_on_submit():  # validate on submit for input by user

        new_transaction=Transaction(account_id=user.id, amount=form.amount_deposit.data,
balance=float((account.balance)+form.amount_deposit.data) , time=datetime.datetime.now(),
type="Deposit") # create new transaction
        account.balance=((account.balance)+form.amount_deposit.data) # update balance
        db.create_all()
        db.session.add(new_transaction)
        db.session.commit()    # commit new transection and updated balance in database
        return '<h1>New depost made!</h1>'

    return render_template('deposit.html', form=form )
    #return '<h1> deposit!</h1>'

# date 16-june-2019     @author RAJ CHHATBAR
@app.route('/withdraw', methods=['GET', 'POST'])
@login_required
def withdraw():
    form=WithdrawForm()   # object for withdraw form
    user = User.query.filter_by(username=current_user.username).first()
    account = Account.query.filter_by(account_id=user.id).first()
    if form.validate_on_submit():
        if (form.amount_withdraw.data)<=account.balance:    # check withdraw less than
balance
            new_transaction=Transaction(account_id=user.id, amount=(-
(form.amount_withdraw.data)), balance=float((account.balance)-
form.amount_withdraw.data) , time=datetime.datetime.now(), type="Withdraw")
            account.balance=((account.balance)-form.amount_withdraw.data)

            db.session.add(new_transaction)
            db.session.commit()
            return '<h1>New withdraw made!</h1>'
        else:
            return '<h1>Insufficient balance!</h1>'   # display insufficient balance

    return render_template('withdraw.html', form=form )
    #return '<h1> withdraw!</h1>'


# date 16-june-2019     @author TAVIN CHOK
```

```
@app.route('/transfer', methods=['GET', 'POST'])    # transfer function
@login_required
def transfer():
    form=TransferForm()   # object for transfer form
    user_sender = User.query.filter_by(username=current_user.username).first()
    account_sender = Account.query.filter_by(account_id=user_sender.id).first()
    if form.validate_on_submit():

        user_recieve = User.query.filter_by(id=form.account_number.data).first()   # recipient
user
        if user_recieve:
            account_recieve = Account.query.filter_by(account_id=user_recieve.id).first()
#recipient account
            if (form.amount_transfer.data)<=account_sender.balance:    # check for transfer
amount less than balance

                new_transaction_debit=Transaction(account_id=user_sender.id, amount=(-
(form.amount_transfer.data)), balance=float((account_sender.balance)-
form.amount_transfer.data) , time=datetime.datetime.now(), type=("Transfer to, {}."
.format(user_recieve.name)) )
# sender user debit transection
                new_transaction_credit=Transaction(account_id=user_recieve.id,
amount=((form.amount_transfer.data)),
balance=float((account_recieve.balance)+form.amount_transfer.data) ,
time=datetime.datetime.now(), type=("Transfer from, {}." .format(user_sender.name))  )
                account_sender.balance=((account_sender.balance)-form.amount_transfer.data)
                account_recieve.balance=((account_recieve.balance)+form.amount_transfer.data)

                db.session.add(new_transaction_debit)
                db.session.add(new_transaction_credit)
                db.session.commit()
                return '<h1>New transfer made!</h1>'
            else:
                return '<h1>Insufficient balance!</h1>'

        else:
            return '<h1>Invalid account number!</h1>'



    return render_template('transfer.html', form=form )

    #return render_template('transfer.html',  )
    #return '<h1> transfer!</h1>'

# date 22-june-2019     @author TAVIN CHOK
```

```
@app.route('/appointment', methods=['GET', 'POST'])
@login_required
def appointment():
    form=AppointmentForm()  # object for appointment form
    user = User.query.filter_by(username=current_user.username).first()
    all_appointment=
Make_an_appointment.query.filter_by(appointment_account_id=user.id).first() # make an
appointment database query for any upcoming appointment
    if form.validate_on_submit():

new_appointment=Make_an_appointment(appointment_account_id=user.id,appointment_dat
e=form.appointment_date.data,appointment_time=form.appointment_time.data,
appointment_location=form.appointment_location.data, about_what=form.about_what.data)
        db.session.add(new_appointment)
        db.session.commit()
        return '<h1>New appointment made!</h1>'



    return render_template('appointment.html',all_appointment=all_appointment , form=form
)
    #return '<h1> appointment!</h1>'




# date 23-june-2019     @author RAJ CHHATBAR
@app.route('/reset', methods=['GET', 'POST'])
def reset_password():
    form=ResetForm()    # reset password form for username
    if form.validate_on_submit():
        user = User.query.filter_by(username=form.reset_username.data).first()
        if user:
            session['forgot_user_name'] = (form.reset_username.data)  # store user name in
session for use in another function

            return redirect(('security_question'))
        else:
            return '<h1>Invalid user!</h1>'

    return render_template('reset.html' , form=form   )

# date 23-june-2019     @author TAVIN CHOK
@app.route('/security_question', methods=['GET', 'POST'])
def reset_user_password():
    form=SecurityForm()  # object for security question answer form
    forgot_user = session.get('forgot_user_name', None)  # get user from session
    user = User.query.filter_by(username=forgot_user).first()
    question_asked=user.security_question
    if form.validate_on_submit():
        que_answer=user.security_question_answer  # question answer from user
        if (str(que_answer)==str(form.question_answer.data)):  # validate user answer
            return redirect(url_for('reset_link'))
        else:
            return "<h1>Invalid answer!</h1>"

    return render_template('security_question.html' ,
question_asked=question_asked,form=form   )

# date 23-june-2019     @author RAJ CHHATBAR
@app.route('/reset_link', methods=['GET', 'POST'])
def reset_link():
    form=ResetPasswordForm()  # object for reset password field
    forgot_user = session.get('forgot_user_name', None)
    user = User.query.filter_by(username=forgot_user).first()
    if form.validate_on_submit():
        hashed_password = generate_password_hash(form.password.data, method='sha256') #
generate new hashed password
        user.password=hashed_password
        db.session.commit()   # commit new password to that user database
        return "<h1>Password changed!</h1>"

    return render_template('reset_link.html' , form=form   )

@app.route('/logout')
@login_required
def logout():   # logot method by using logout_user() prebuild method
    logout_user()
    return redirect(url_for('index'))

if __name__ == '__main__':
    db.create_all()  # creare database table
    app.run(debug=True)  # run system in debug mode
```

////////////////////////////This is front end of software////////////////////

Withdraw.html

```
{% extends "bootstrap/base.html" %}
{% import "bootstrap/wtf.html" as wtf %}

{% block title %}
Dashboard
{% endblock %}

{% block styles %}
{{super()}}
<link rel="stylesheet" href="{{url_for('.static', filename='dashboard.css')}}">
{% endblock %}

{% block content %}
    <nav class="navbar navbar-inverse navbar-fixed-top">
      <div class="container-fluid">
        <div class="navbar-header">
          <button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-target="#navbar" aria-expanded="false" aria-controls="navbar">
            <span class="sr-only">Toggle navigation</span>
            <span class="icon-bar"></span>
            <span class="icon-bar"></span>
            <span class="icon-bar"></span>
          </button>
          <a class="navbar-brand" href="{{ url_for('dashboard') }}">Bank of Squardo</a>
        </div>
        <div id="navbar" class="navbar-collapse collapse">  // Setting up navbar
          <ul class="nav navbar-nav navbar-right">
            <li><a href="#">Settings</a></li>
            <li><a href="#">Profile</a></li>
            <li><a href="{{ url_for('logout') }}">Log Out</a></li>  // connect backend url for function logout
          </ul>
          <!--
          <form class="navbar-form navbar-right">
            <input type="text" class="form-control" placeholder="Search...">
          </form>
        -->
        </div>
      </div>
    </nav>

    <div class="container-fluid">
      <div class="row">
        <div class="col-sm-3 col-md-2 sidebar">
          <ul class="nav nav-sidebar">
            <li ><a href="{{ url_for('dashboard') }}">Overview <span // connect backend url for
function dashboard
class="sr-only">(current)</span></a></li>
            <li ><a href="{{ url_for('deposit') }}">Deposit</a></li>// connect backend url for
function deposit

            <li class="active" ><a href="{{ url_for('withdraw') }}">Withdraw</a></li>
            <li><a href="{{ url_for('transfer') }}">Transfer</a></li>
// connect backend url for function transfer

            <li><a href="{{ url_for('appointment') }}">Make an appointment</a></li>// connect
backend url for function appoinment

          </ul>



        </div>
        <div class="col-sm-9 col-sm-offset-3 col-md-10 col-md-offset-2 main">
          <h1 class="page-header">Tell us how much you want to withdraw today!</h1>

          <form class="form-signin" method="POST" action="/withdraw" >
            {{ form.hidden_tag() }}
            {{form.amount_withdraw.label}} {{ form.amount_withdraw }}// connect backend
values for withdraw amount and submit button

             <button class="btn btn-lg btn-primary btn-block"
type="submit">Withdraw</button>
          </form>





        </div>
      </div>
    </div>
{% endblock %}
```

Transfer.html

```
{% extends "bootstrap/base.html" %}
{% import "bootstrap/wtf.html" as wtf %}

{% block title %}
Dashboard
{% endblock %}

{% block styles %}
{{super()}}
<link rel="stylesheet" href="{{url_for('.static', filename='dashboard.css')}}">
{% endblock %}

{% block content %}
    <nav class="navbar navbar-inverse navbar-fixed-top">
      <div class="container-fluid">
        <div class="navbar-header">
          <button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-target="#navbar" aria-expanded="false" aria-controls="navbar">
            <span class="sr-only">Toggle navigation</span>
            <span class="icon-bar"></span>
            <span class="icon-bar"></span>
            <span class="icon-bar"></span>
          </button>
          <a class="navbar-brand" href="{{ url_for('dashboard') }}">Bank of Squardo</a>
        </div>
        <div id="navbar" class="navbar-collapse collapse">
          <ul class="nav navbar-nav navbar-right">
            <li><a href="#">Settings</a></li>
            <li><a href="#">Profile</a></li>
            <li><a href="{{ url_for('logout') }}">Log Out</a></li>
          </ul>
          <!--
          <form class="navbar-form navbar-right">
            <input type="text" class="form-control" placeholder="Search...">
          </form>
        -->
        </div>
      </div>
    </nav>

    <div class="container-fluid">
      <div class="row">
        <div class="col-sm-3 col-md-2 sidebar">
```

```
          <ul class="nav nav-sidebar">
            <li ><a href="{{ url_for('dashboard') }}">Overview <span class="sr-only">(current)</span></a></li>
            <li ><a href="{{ url_for('deposit') }}">Deposit</a></li>
            <li  ><a href="{{ url_for('withdraw') }}">Withdraw</a></li>
            <li class="active"><a href="{{ url_for('transfer') }}">Transfer</a></li>
            <li><a href="{{ url_for('appointment') }}">Make an appointment</a></li>
          </ul>


        </div>
        <div class="col-sm-9 col-sm-offset-3 col-md-10 col-md-offset-2 main">
          <h1 class="page-header">Tell us how much you want to transfer today!</h1>

          <form class="form-signin" method="POST" action="/transfer" >
            {{ form.hidden_tag() }}
            {{form.amount_transfer.label}} {{ form.amount_transfer }}
            {{form.account_number.label}} {{ form.account_number }}
             <button class="btn btn-lg btn-primary btn-block" type="submit">Transfer</button>
          </form>




        </div>
      </div>
    </div>
{% endblock %}
```

Signup.html

```
{% extends "bootstrap/base.html" %}
{% import "bootstrap/wtf.html" as wtf %}

{% block title %}
<h1>Bank of squardo</h1>
Sign Up
{% endblock %}

{% block styles %}
{{super()}}
<link rel="stylesheet" href="{{url_for('.static', filename='signin.css')}}">
{% endblock %}

{% block content %}
<nav class="navbar navbar-inverse navbar-fixed-top">
```

```
  <div class="container">
    <div class="navbar-header">
      <button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-target="#navbar" aria-expanded="false" aria-controls="navbar">
        <span class="sr-only">Toggle navigation</span>
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
      </button>
      <a class="navbar-brand" href="#">Bank of Squardo</a>
    </div>
    <div id="navbar" class="collapse navbar-collapse">
      <ul class="nav navbar-nav">
        <li ><a href="{{ url_for('index') }}">Home</a></li>
        <li ><a href="{{ url_for('login') }}">Login</a></li>
        <li class="active"><a href="{{ url_for('signup') }}">Sign Up</a></li>
      </ul>
    </div><!--/.nav-collapse -->
  </div>
</nav>

<div class="container">

  <form class="form-signin" method="POST" action="/signup">
    <h2 class="form-signin-heading">Sign Up</h2>
    {{ form.hidden_tag() }}
    {{ wtf.form_field(form.username) }}
    {{ wtf.form_field(form.email) }}
    {{ wtf.form_field(form.name) }}
    {{ wtf.form_field(form.gender) }}
    {{ wtf.form_field(form.address) }}
    {{ wtf.form_field(form.city) }}
    {{ wtf.form_field(form.state) }}
    {{ wtf.form_field(form.zipcode) }}
    {{ wtf.form_field(form.date_of_birth) }}
    {{ wtf.form_field(form.phone_number) }}
    {{ wtf.form_field(form.password) }}
    {{ wtf.form_field(form.security_question) }}
    {{ wtf.form_field(form.security_question_answer) }}
    <button class="btn btn-lg btn-primary btn-block" type="submit">Sign Up</button>
  </form>

</div> <!-- /container -->
{% endblock %}
```

Security_question.html

```
{% extends "bootstrap/base.html" %}
{% import "bootstrap/wtf.html" as wtf %}

{% block title %}

{% endblock %}

{% block styles %}
{{super()}}
<link rel="stylesheet" href="{{url_for('.static', filename='signin.css')}}">
{% endblock %}

{% block content %}

    <div class="container">



  <form class="form-signin" method="POST" action="security_question">
      <h3 class="fs-subtitle">step 2</h3>
        <h2 class="form-signin-heading">  {{ question_asked }}</h2>


  {{ form.hidden_tag() }}

  {{ wtf.form_field(form.question_answer) }}

  <button class="btn btn-lg btn-primary btn-block" type="submit">Next</button>





      </form>

    </div> <!-- /container -->
{% endblock %}
```

Reset_link.html

```
{% extends "bootstrap/base.html" %}
{% import "bootstrap/wtf.html" as wtf %}

{% block title %}
```

```
Login
{% endblock %}

{% block styles %}
{{super()}}
<link rel="stylesheet" href="{{url_for('.static', filename='signin.css')}}">
{% endblock %}

{% block content %}

    <div class="container">

      <form class="form-signin"  method="POST" action="/reset_link">
          <h3 class="fs-subtitle">step 3</h3>
        <h2 class="form-signin-heading">Enter new password</h2>

        <fieldset>


  {{ form.hidden_tag() }}
  {{ wtf.form_field(form.password) }}

  <button class="btn btn-lg btn-primary btn-block" type="submit">Reset</button>

</fieldset>



      </form>

    </div> <!-- /container -->
{% endblock %}
```

Reset.html

```
{% extends "bootstrap/base.html" %}
{% import "bootstrap/wtf.html" as wtf %}

{% block title %}
Login
{% endblock %}

{% block styles %}
{{super()}}
<link rel="stylesheet" href="{{url_for('.static', filename='signin.css')}}">
{% endblock %}
{% block content %}

    <div class="container">

      <form class="form-signin" method="POST" action="reset" >

        <h2 class="form-signin-heading">Reset password</h2>

        <fieldset>

  <h3 class="fs-subtitle">step 1</h3>
  {{ form.hidden_tag() }}
  {{ wtf.form_field(form.reset_username) }}

  <button class="btn btn-lg btn-primary btn-block" type="submit">Next</button>

</fieldset>



      </form>

    </div> <!-- /container -->
{% endblock %}
```

Login.html

```
{% extends "bootstrap/base.html" %}
{% import "bootstrap/wtf.html" as wtf %}

{% block title %}
Login
{% endblock %}

{% block styles %}
{{super()}}
<link rel="stylesheet" href="{{url_for('.static', filename='signin.css')}}">
{% endblock %}

{% block content %}
<nav class="navbar navbar-inverse navbar-fixed-top">
  <div class="container">
    <div class="navbar-header">
      <button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-target="#navbar" aria-expanded="false" aria-controls="navbar">
        <span class="sr-only">Toggle navigation</span>
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
      </button>
      <a class="navbar-brand" href="#">Bank of Squardo</a>
    </div>
    <div id="navbar" class="collapse navbar-collapse">
      <ul class="nav navbar-nav">
        <li ><a href="{{ url_for('index') }}">Home</a></li>
        <li class="active"><a href="{{ url_for('login') }}">Login</a></li>
        <li><a href="{{ url_for('signup') }}">Sign Up</a></li>
      </ul>
    </div><!--/.nav-collapse -->
  </div>
</nav>
    <div class="container">

      <form class="form-signin" method="POST" action="/login">
        <h2 class="form-signin-heading">Please sign in</h2>
        {{ form.hidden_tag() }}
        {{ wtf.form_field(form.username) }}
        {{ wtf.form_field(form.password) }}
        {{ wtf.form_field(form.remember) }}

        <button class="btn btn-lg btn-primary btn-block" type="submit">Sign in</button>
        <p></p>
  <a  class="form-signin-heading"   href="{{ url_for('reset_password') }}">reset
password</a>
      </form>


    </div> <!-- /container -->
{% endblock %}
```

Index.html

```
{% extends "bootstrap/base.html" %}

{% block title %}
Bank of squardo
{% endblock %}

{% block styles %}
{{super()}}
<link rel="stylesheet" href="{{url_for('.static', filename='starter-template.css')}}">
{% endblock %}
{% block content %}
    <nav class="navbar navbar-inverse navbar-fixed-top">
      <div class="container">
        <div class="navbar-header">
          <button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-target="#navbar" aria-expanded="false" aria-controls="navbar">
            <span class="sr-only">Toggle navigation</span>
            <span class="icon-bar"></span>
            <span class="icon-bar"></span>
            <span class="icon-bar"></span>
          </button>
          <a class="navbar-brand" href="#">Bank of Squardo</a>
        </div>
        <div id="navbar" class="collapse navbar-collapse">
          <ul class="nav navbar-nav">
            <li class="active"><a href="#">Home</a></li>
            <li><a href="{{ url_for('login') }}">Login</a></li>
            <li><a href="{{ url_for('signup') }}">Sign Up</a></li>
          </ul>
        </div><!--/.nav-collapse -->
      </div>
    </nav>

    <div class="container">

      <div class="starter-template">
        <h1>Bank of squardo</h1>
        <p class="lead">Use this document as a way to quickly start any new project.<br> All you get is this text and a mostly barebones HTML document.</p>
      </div>

    </div><!-- /.container -->
{% endblock %}
```

Deposit.html

```
{% extends "bootstrap/base.html" %}
{% import "bootstrap/wtf.html" as wtf %}

{% block title %}
Dashboard
{% endblock %}

{% block styles %}
{{super()}}
<link rel="stylesheet" href="{{url_for('.static', filename='dashboard.css')}}">
```

```
{% endblock %}

{% block content %}
    <nav class="navbar navbar-inverse navbar-fixed-top">
      <div class="container-fluid">
        <div class="navbar-header">
          <button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-target="#navbar" aria-expanded="false" aria-controls="navbar">
            <span class="sr-only">Toggle navigation</span>
            <span class="icon-bar"></span>
            <span class="icon-bar"></span>
            <span class="icon-bar"></span>
          </button>
          <a class="navbar-brand" href="{{ url_for('dashboard') }}">Bank of Squardo</a>
        </div>
        <div id="navbar" class="navbar-collapse collapse">
          <ul class="nav navbar-nav navbar-right">
            <li><a href="#">Settings</a></li>
            <li><a href="#">Profile</a></li>
            <li><a href="{{ url_for('logout') }}">Log Out</a></li>
          </ul>
          <!--
          <form class="navbar-form navbar-right">
            <input type="text" class="form-control" placeholder="Search...">
          </form>
        -->
        </div>
      </div>
    </nav>

    <div class="container-fluid">
      <div class="row">
        <div class="col-sm-3 col-md-2 sidebar">
          <ul class="nav nav-sidebar">
            <li ><a href="{{ url_for('dashboard') }}">Overview <span class="sr-only">(current)</span></a></li>
            <li class="active"><a href="{{ url_for('deposit') }}">Deposit</a></li>
            <li><a href="{{ url_for('withdraw') }}">Withdraw</a></li>
            <li><a href="{{ url_for('transfer') }}">Transfer</a></li>
            <li><a href="{{ url_for('appointment') }}">Make an appointment</a></li>
          </ul>


        </div>
        <div class="col-sm-9 col-sm-offset-3 col-md-10 col-md-offset-2 main">
          <h1 class="page-header">Tell us how much you want to deposit today!</h1>
```

```
     <form class="form-signin" method="POST" action="/deposit" >
      {{ form.hidden_tag() }}
      {{form.amount_deposit.label}} {{ form.amount_deposit }}
       <button class="btn btn-lg btn-primary btn-block" type="submit">Deposit</button>
     </form>




    </div>
   </div>
  </div>
{% endblock %}
```

Dashboard.html

```
{% extends "bootstrap/base.html" %}

{% block title %}
Dashboard
{% endblock %}

{% block styles %}
{{super()}}
<link rel="stylesheet" href="{{url_for('.static', filename='dashboard.css')}}">
{% endblock %}

{% block content %}
  <nav class="navbar navbar-inverse navbar-fixed-top">
   <div class="container-fluid">
    <div class="navbar-header">
     <button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-target="#navbar" aria-expanded="false" aria-controls="navbar">
      <span class="sr-only">Toggle navigation</span>
      <span class="icon-bar"></span>
      <span class="icon-bar"></span>
      <span class="icon-bar"></span>
     </button>
     <a class="navbar-brand" href="{{ url_for('dashboard') }}">Bank of Squardo</a>
    </div>
    <div id="navbar" class="navbar-collapse collapse">
     <ul class="nav navbar-nav navbar-right">
      <li><a href="#">Settings</a></li>
      <li><a href="#">Profile</a></li>
```

```
            <li><a href="{{ url_for('logout') }}">Log Out</a></li>
          </ul>
          <!--
          <form class="navbar-form navbar-right">
            <input type="text" class="form-control" placeholder="Search...">
          </form>
        -->
        </div>
      </div>
    </nav>

    <div class="container-fluid">
      <div class="row">
        <div class="col-sm-3 col-md-2 sidebar">
          <ul class="nav nav-sidebar">
            <li class="active"><a href="#">Overview <span class="sr-only">(current)</span></a></li>
            <li><a href="{{ url_for('deposit') }}">Deposit</a></li>
            <li><a href="{{ url_for('withdraw') }}">Withdraw</a></li>
            <li><a href="{{ url_for('transfer') }}">Transfer</a></li>
            <li><a href="{{ url_for('appointment') }}">Make an appointment</a></li>
          </ul>


        </div>
        <div class="col-sm-9 col-sm-offset-3 col-md-10 col-md-offset-2 main">
          <h1 class="page-header">Welcome to Bank of Squardo, {{ user_name }}</h1>

          <div class="row placeholders">

            <div class="col-xs-6 col-sm-3 placeholder">
              <img src="data:image/gif;base64,R0lGODlhAQABAIAAAHd3dwAAACH5BAAAAAAALAAAAAABAAEAAAICRAEAOw==" width="200" height="200" class="img-responsive" alt="Generic placeholder thumbnail">
              <h4>Photo</h4>
              <span class="text-muted">Something else</span>
            </div>

            <div class="col-xs-6 col-sm-3 placeholder">

                Account No.: <h4>{{ id }}</h4>
                Name:<h4>{{ name }}</h4>

                Address:<h4>{{ address }}</h4>
                City:<h4>{{ city }}</h4>
                State:<h4>{{ state }}</h4>
                Zip-code:<h4>{{ zipcode }}</h4>
                Phone No.:<h4>{{ phone }}</h4>

            </div>


            <div class="col-xs-6 col-sm-3 placeholder">

                Balance:<h4>$ {{ balance }}</h4>

            </div>

          </div>

          <h2 class="sub-header">Transactions</h2>
          <div class="table-responsive">
            <table class="table table-striped">
              <thead>
                <tr>
                  <th>Transaction id</th>
                  <th>Amount</th>
                  <th>Balance</th>
                  <th>Time</th>
                  <th>Type</th>
                </tr>
              </thead>
              <tbody>


                {% for transaction in all_transactions | reverse %}
                <tr>
                  <td>{{transaction.transfer_id}}</td>
                  <td>$ {{transaction.amount}}</td>
                  <td>$ {{transaction.balance}}</td>
                  <td>{{transaction.time}}</td>
                  <td>{{transaction.type}}</td>
                </tr>


                {% endfor %}


              </tbody>
            </table>
          </div>
```

```
        </div>
      </div>
    </div>
{% endblock %}
```

Appointment.html

```
{% extends "bootstrap/base.html" %}
{% import "bootstrap/wtf.html" as wtf %}

{% block title %}
Dashboard
{% endblock %}

{% block styles %}
{{super()}}
<link rel="stylesheet" href="{{url_for('.static', filename='dashboard.css')}}">
{% endblock %}

{% block content %}
    <nav class="navbar navbar-inverse navbar-fixed-top">
      <div class="container-fluid">
        <div class="navbar-header">
          <button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-target="#navbar" aria-expanded="false" aria-controls="navbar">
            <span class="sr-only">Toggle navigation</span>
            <span class="icon-bar"></span>
            <span class="icon-bar"></span>
            <span class="icon-bar"></span>
          </button>
          <a class="navbar-brand" href="{{ url_for('dashboard') }}">Bank of Squardo</a>
        </div>
        <div id="navbar" class="navbar-collapse collapse">
          <ul class="nav navbar-nav navbar-right">
            <li><a href="#">Settings</a></li>
            <li><a href="#">Profile</a></li>
            <li><a href="{{ url_for('logout') }}">Log Out</a></li>
          </ul>
          <!--
          <form class="navbar-form navbar-right">
            <input type="text" class="form-control" placeholder="Search...">
          </form>
        -->
        </div>
      </div>
    </nav>

    <div class="container-fluid">
      <div class="row">
        <div class="col-sm-3 col-md-2 sidebar">
          <ul class="nav nav-sidebar">
            <li ><a href="{{ url_for('dashboard') }}">Overview <span class="sr-
only">(current)</span></a></li>
            <li ><a href="{{ url_for('deposit') }}">Deposit</a></li>
            <li><a href="{{ url_for('withdraw') }}">Withdraw</a></li>
            <li><a href="{{ url_for('transfer') }}">Transfer</a></li>
            <li class="active"><a href="{{ url_for('appointment') }}">Make an
appointment</a></li>
          </ul>


        </div>
        <div class="col-sm-9 col-sm-offset-3 col-md-10 col-md-offset-2 main">
          <h1 class="page-header">Let's talk about...</h1>

          <form class="form-signin" method="POST" action="/appointment" >
            {{ form.hidden_tag() }}
            {{form.appointment_date.label}} {{ form.appointment_date }}
            {{form.appointment_time.label}} {{ form.appointment_time }}
            {{form.appointment_location.label}} {{ form.appointment_location }}
            {{form.about_what.label}} {{ form.about_what }}

             <button class="btn btn-lg btn-primary btn-block" type="submit">Schedule
it!</button>
          </form>

          <h2 class="sub-header">Upcoming appointment</h2>

          <div class="table-responsive">
            <table class="table table-striped">
              <thead>
                <tr>
                  <th>Appointment date</th>
                  <th>Appointment time</th>
                  <th>Appointment Location </th>
                  <th>About what?</th>

                </tr>
              </thead>
              <tbody>
```

```
                <tr>
                  <td>{{all_appointment.appointment_date}}</td>
                  <td>{{all_appointment.appointment_time}}</td>
                  <td> {{all_appointment.appointment_location}}</td>
                  <td>{{all_appointment.about_what}}</td>

                </tr>





              </tbody>
            </table>
          </div>



        </div>
      </div>
    </div>
{% endblock %}
```