Early Detection of Chronic Kidney Disease using Machine Learning

HX8001

Professional Readiness for Innovation, Employability and Entrepreneurship

PROJECT REPORT

Submitted by Team ID: PNT2022TMID26456

SWETHA S	**211719104149**
SIVARANJANI M G	**211719104133**
SOWMIYA E	**211719104134**
SRINIDHI R	**211719104136**
SWETHA J K	**211719104148**

in partial fulfilment for the award of the

degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

RAJALAKSHMI INSTITUTE OF TECHNOLOGY

ANNA UNIVERSITY: CHENNAI 600 025

NOVEMBER 2022

BONAFIDE CERTIFICATE

Certified that this project report **"EARLY DETECTION OF CHRONIC KIDNEY DISEASE USING MACHINE LEARNING"** is the bonafide work of SWETHA.S(211719104149),SIVARANJANI.M.G(211719104133),SOWMIYA.E(211719104134),SRINIDHI.R(211719104136),SWETHAJ.K (211719104148) who carried out the project work under my supervision.

SIGNATURE:

Dr. T.RAJENDRAN,

M.E., Ph.D,

HEAD OF THE DEPARTMENT,

Dept. of Computer Science and Engg.,

Rajalakshmi Institute of

Technology,

Kuthambakkam Post,

Chennai - 600 124

SIGNATURE:

Mrs. SUBHA.S,

M.TECH., Ph.D,

MENTOR,

Dept. of Computer Science

Rajalakshmi Institute of

Technology,

Kuthambakkam Post,

Chennai - 600 124

The viva-voce is held on_____.

INTERNAL EXAMINER **EXTERNAL EXAMINER**

1. INTRODUCTION

1.1 Project Overview:

Chronic Kidney Disease (CKD) is a long-term condition in which the loss of kidney function occurs in a slow and progressive manner. The kidneys lose their ability to filter out the waste and fluid from the blood. The main risk factors for developing kidney disease are diabetes, high blood pressure, heart disease, and a family history of kidney failure. Unfortunately, CKD is an irreversible disease. It progresses to successive stages. It has five stages. In the early stages of chronic kidney disease, you might have few signs or symptoms. One might not realize the kidney damage until the condition is advanced. In stage 1, kidneys function normally and the GFR levels are normal but presence of protein is found in the urine which indicates CKD. If detected earlier, its advancement to the next stage can be prevented or delayed. Diabetes and high blood pressure are the most common causes of kidney disease. The cause of your kidney disease may affect the type of treatment you receive. CKD is also for some people asymptomatic or the symptoms are not specific to the disease, therefore making it difficult to predict it. This is why it is important for people to take up regular health check-ups and look out for any symptoms that might be an indication to CKD. Getting tested regularly may be the only way to predict CKD. The earlier it is diagnosed, the more effectual the treatment is.

The doctors will usually perform certain tests to find out whether the patient has been affected with CKD, and if yes, then what stage of CKD is the patient going through is found to know the severity and provide treatment accordingly. The treatment may include medications and in advanced stages, dialysis.

In this model, Random Forest Algorithm is used to predict the presence of CKD with higher accuracy. The features are reduced, for the improvement of accuracy level, by eliminating certain features using Recursive Feature Elimination (RFE).

1.2 Purpose:

The main purpose of the model is to predict chronic kidney disease with high accuracy for patients to get treatment accordingly.

2. LITERATURE SURVEY

2.1 Existing Problem:

Chronic Kidney Disease (CKD) is one of the deadliest diseases that slowly damages human kidney. The disease remains undetected in its early stage and the patients can only realize the severity of the disease when it gets advanced. Hence, detecting such disease at earlier stage isa key challenge now. Machine Learning is one of the emerging fields used in the health sectorsfor the diagnosis of different diseases. In this model, we compute, analyze and compare between Machine Learning classification approaches to determine which classification approach is the optimal for the prediction of CKD. Some renowned machine learning methods were selected to train the model and based on these results, we can compare and determine which among the following Machine Learning Methods and predict the possibility of CKD at the most accurate level. Among these, Random Forest algorithm proves to be of high accuracy.

RANDOM FOREST ALGORITHM

Random Forest algorithm is a classification method that contains several decision trees on various subsets of a given dataset and takes the average to enhance the predicted accuracy. We will be using Recursive Feature Elimination (RFE) to reduce the features to improve accuracy.

RECURSIVE FEATURE ELIMINATION

Recursive Feature Elimination (RFE) is a technique used to eliminate the features that are ranked to be the least important. Using this, the least important features are reduced and only the features that marks to be of utmost importance are taken into account. This will improve the level of accuracy, hence making the prediction more accurate.

2.2 REFERENCES

1. Jager K.J., Kovesday C., Langham R., et al. A single number for advocacy and communication-worldwide more than 850 million individuals have kidney diseases. Kidney Int. 2019; 96:1048-1050

2. A Machine Learning Methodology for Diagnosing Chronic Kidney Disease by Jiongmin Qin, LinChen, Yuhua Liu, Chuanjun Liu, Changhao Feng, Bin Chen

3. Diagnostic decision support system of chronic kidney disease using support vector machine Mubarik Ahmad, Vitri Tunjungsari, Dini Widianti, Peny Amalia, Ummi Azizah Rachmawati.

4. Salekin, A., & Stankovic, J. (2016). Detection of Chronic Kidney Disease and Selecting Important Predicitive Attributes.

5. Performance Evaluation on Machine Learning Classification Techniques for Disease Classification and Forecasting through Data Analytics for Chronic Kidney Disease (CKD) Gunarathne W.H.S.D, Perera K.D.M, Kahandawaarachchi K.A.D.

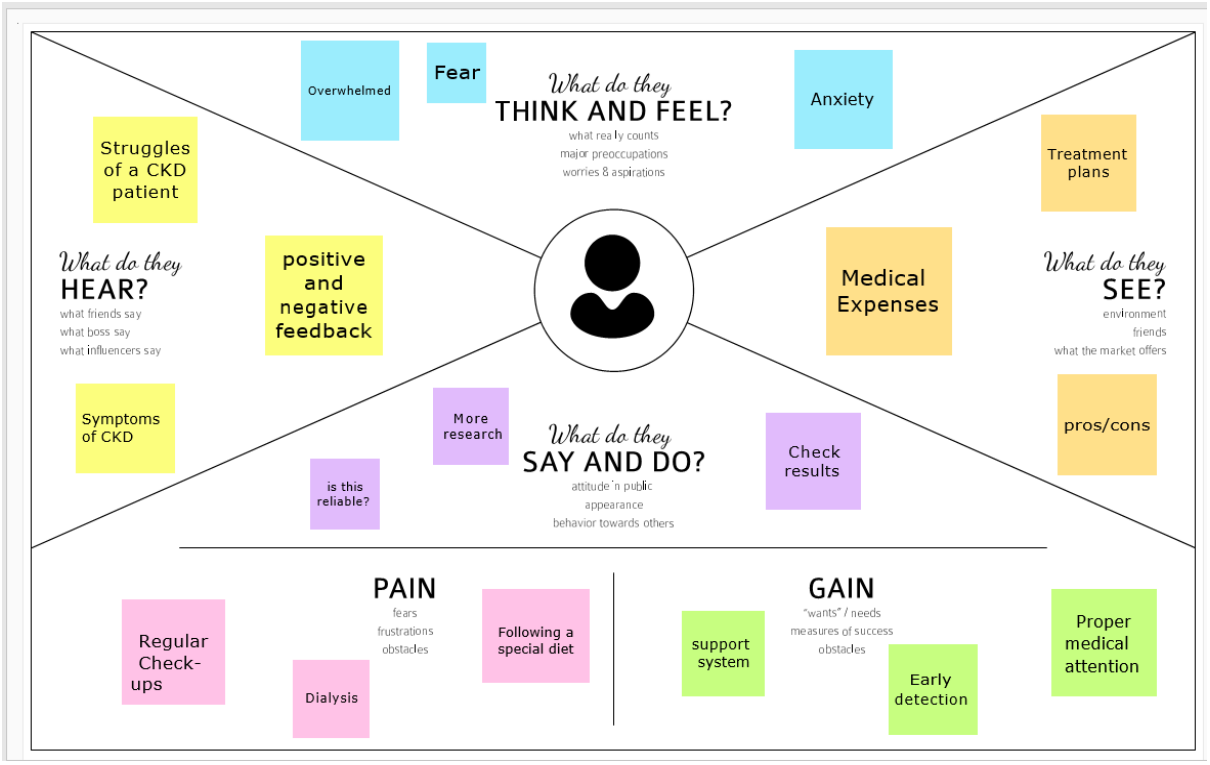
2.3 PROBLEM STATEMENT DEFINITION

Chronic Kidney Disease is an increasing issue in today's world and most of the people who are affected by this, get to know that they have CKD only in later stages as it is difficult to predict it in its early stages. People who experience symptoms of CKD, must regularly get tested so that they can predict it earlier. Predicting CKD earlier can help the patients to slow down the progression of kidney damage and stay as healthy as possible.

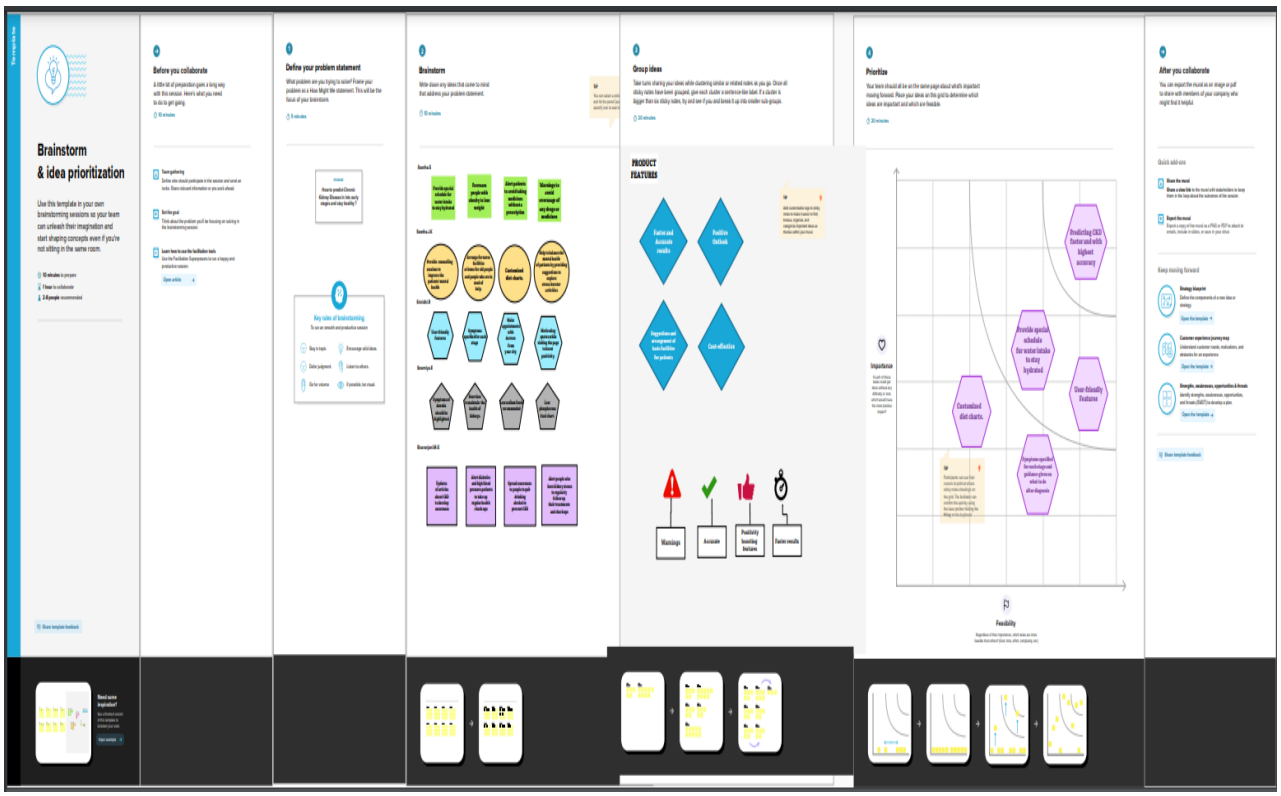
This model aims to predict the presence of CKD accurately so that the patients can get awareness about their kidneys' condition.

3.IDEATION AND PROPOSED SOLUTION

3.1 Empathy Map Canvas:



3.2 Ideation and Brainstorming:



3.3 Proposed Solution:

S. No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	Chronic Kidney Disease (CKD) is a progressive disease that has become a global health crisis. A model that would help to predict this disease in its early stages will be effective in halting or delaying its progression by providing the necessary treatment to the patients.
2.	Idea / Solution description	**Steps to be performed:** Pre-processing or cleaning of the data sets. Next, analysing the pre-processed data. Train the machine with the pre-processed data. We use and compare different algorithms and choose the one that proves to be highly accurate.
3.	Novelty / Uniqueness	The key indicators of CKD are eGFR levels and urine albumin. GFR level aberrations will be taken into account for predicting CKD. Eliminating features that provide very less contribution to detect the kidney disease and using important features would provide results that are more accurate.

4.	Social Impact / Customer Satisfaction	The objective of this system is to predict kidney disease earlier. Advanced stage treatment might require dialysis which is expensive and most of the people will not be able to afford it. Early prediction will help in providing the needed treatment to prevent the disease from advancing to the next stage.
5.	Business Model (Revenue Model)	Profits can be gained by collaborating this model with the healthcare sectors and MNCs. People who undergo CKD prediction tests will get accurate results according to which the hospital can treat them. It can facilitate better quality care for chronic illness and hence can be beneficial for the healthcare sectors as well.
6.	Scalability of the Solution	The proposed model to predict CKD is best suited for handling larger datasets.

3.4 Problem Solution Fit:

1. CUSTOMER SEGMENT(S)	6. CUSTOMER CONSTRAINTS	5. AVAILABLE SOLUTIONS
People who have symptoms of CKD and people who want to get their kidneys checked for staying healthy and fit. Doctors and workers in the healthcare sectors.	The test and treatments for advanced stages are highly priced. The patients have to wait for a longer time to get their test results which may cause more complications in the meantime.	Proper medications Regime diets Dialysis.

2. JOBS-TO-BE-DONE / PROBLEMS:	9. PROBLEM ROOT	7. BEHAVIOUR
Create an interface that is convenient for the users to operate and easy to understand. Ensure that the predictions are of high accuracy. Provide accurate and faster results by using the given data set so that the kidney disease can be detected earlier.	Ignorance of symptoms and lack of awareness about the disease. Unhealthy diet. Not having enough water. Consumption of alcohol or tobacco frequently.	Notice the changes in your body and lookout for the symptoms of CKD. Consult the doctors if you notice CKD symptoms. Develop awareness and be cautious about the prevent disease. do not ignore the symptoms or be careless about those.

3. TRIGGERS	10. YOUR SOLUTION	8. CHANNELS OF BEHAVIOUR
Usually, CKD tests takes longer time to predict the disease. These tests are expensive and are not easily affordable by many people.	Building a machine learning model that will predict CKD in its early stages by providing accurate and faster results.	**ONLINE:** Consult a specialist online to follow the necessary medications. Browse about the disease to be cautious.
4. EMOTIONS: BEFORE / AFTER People are comfused and worried before taking up the prediction test. As they don't know why there are having the symptoms, they don't get proper treatment until they find out that they symptoms are leading to CKD. Without proper medications, the chances of complications are high. After detecting the disease, patients gain more clarity and can be directed towards getting the proper medication. Hence, they can be more hopeful.		**OFFLINE:** Get dialysis and kidney transplations for advanced stage and medications for initial stages.

4. REQUIREMENT ANALYSIS

4.1 Functional Requirements:

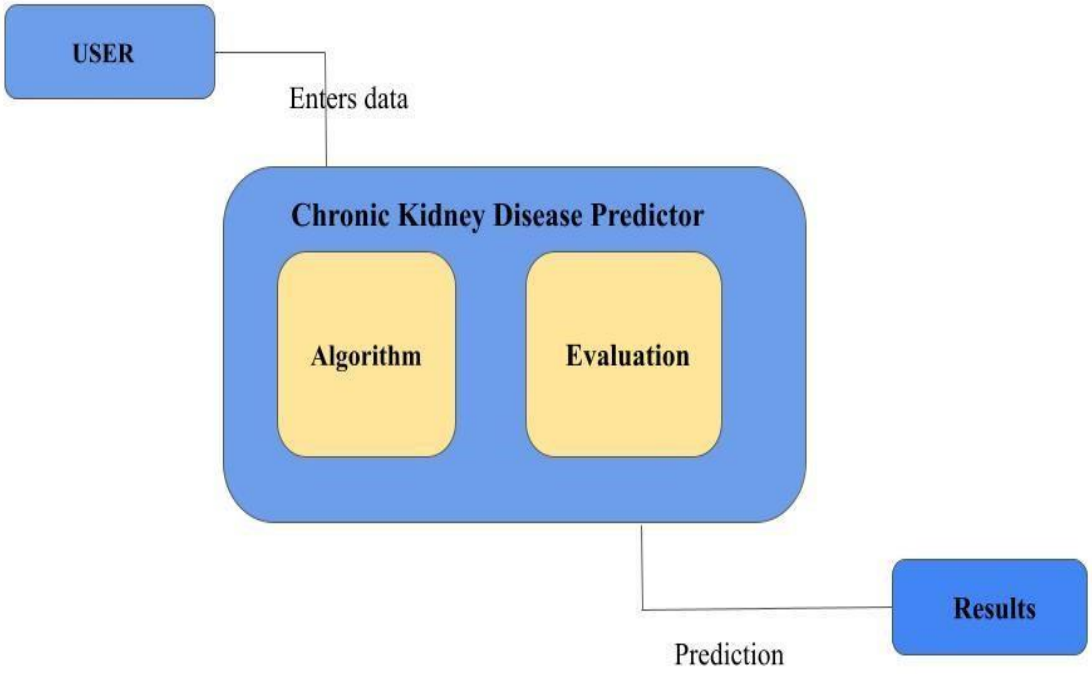
FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	This system allows user to register through Form.
FR-2	User Confirmation	Confirmation via Email.
FR-3	Reset Password.	This system allows users to change their password.
FR-4	Input Data	This system must allow users to enter data for prediction.
FR-5	Prediction	This system must predict kidney disease
FR-6	Feedback	This system must allow users to give feedback by taking a survey.

4.2Non-Functional Requirements:

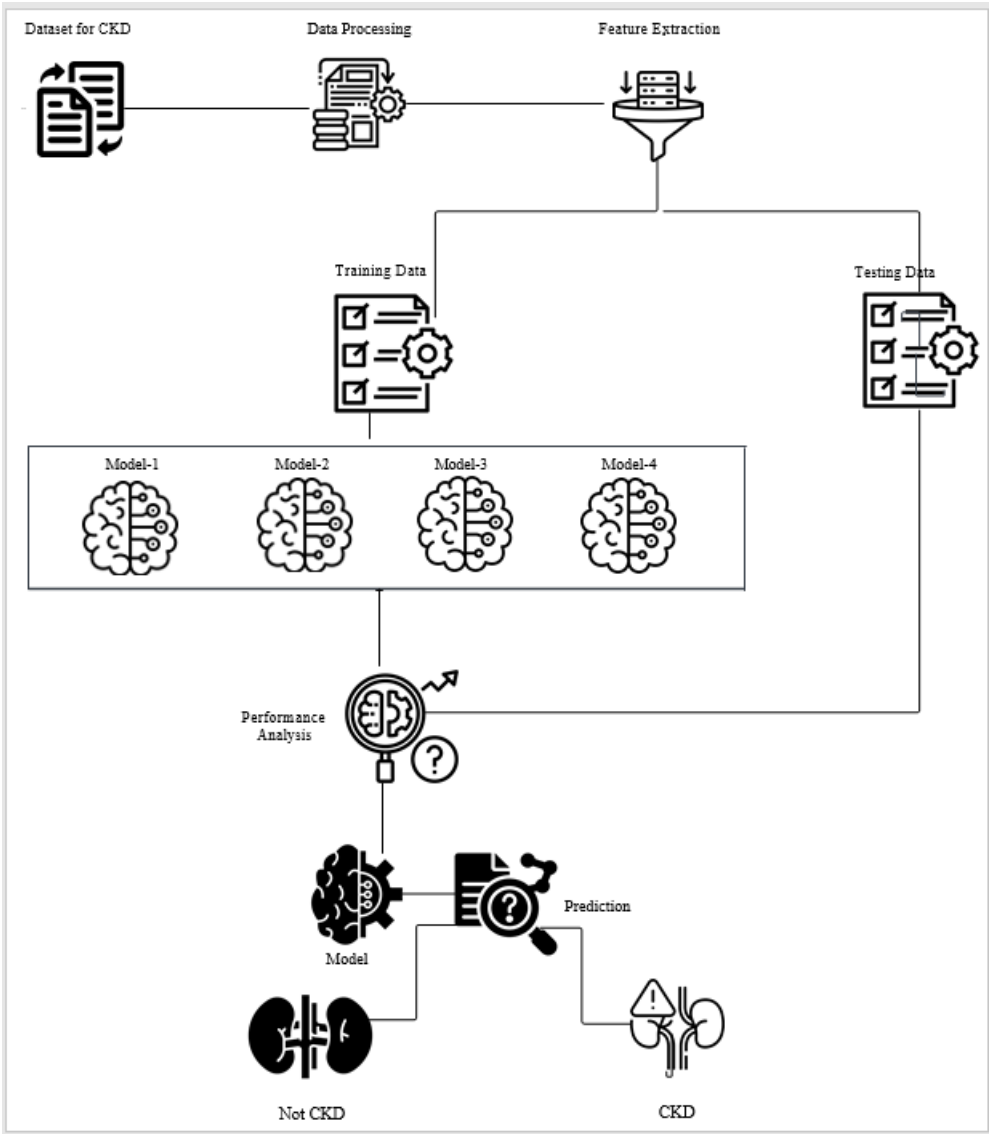
FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	This system should have user-friendly features.
NFR-2	**Security**	This system should protect user data and allow only authorized persons to access the data.
NFR-3	**Reliability**	This system should have low failure rate.
NFR-4	**Performance**	This system should take less time to predict the results.
NFR-5	**Availability**	This system should be able to perform on all environment.
NFR-6	**Scalability**	This system should manage high workloads.

5.PROJECT DESIGN

5.1 Data Flow Diagram:



5.2 Solution & Technical Architecture:



5.3 User Stories:

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (web user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	High	Sprint-1
		USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	High	Sprint-1
		USN-3	As a user, I can register for the application through social media.	I can register & access the dashboard with social media Login	Low	Sprint-4
		USN-4	As a user, I can register for the application through Gmail		Medium	Sprint-2
	Login	USN-5	As a user, I can log into the application by entering email & password		High	Sprint-1
	Dashboard	USN-6	As a user, I can view further measures regarding my results.		High	Sprint-3
Customer (mobile user)	Password	USN-1	As a user, I can change my password.		High	Sprint-1
	Prediction	USN-2	As a user, I can see my results by entering the data		High	Sprint-2
Admin	Data Cleaning	USN-1	Clean the dataset for processing.		High	Sprint-2
	Model building	USN-1	Choosing the effective algorithm for prediction.		High	Sprint-2

6. PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning and Estimation:

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	7	High	Swetha JK, Sivaranjani M G, Sowmiya E
Sprint-1		USN-2	As a user, I will receive confirmation email once I have registered for the application	7	High	Swetha s
Sprint-1		USN-3	Reset password	6	Medium	Srinidhi R
Sprint-1		USN-4	As a user, I can register for the application through Gmail	2	Medium	Srinidhi R
Sprint-1	Login	USN-5	As a user, I can log into the application by entering email & password	7	High	Swetha JK, Sivaranjani M G, Sowmiya E
Sprint-3	Dashboard	USN-6	As a user, I can view further measures regarding my results.	10	High	Sowmiya E, Swetha S, Srinidhi R
Sprint-3	Prediction	USN-2	As a user, I can see my results by entering the data	10	High	Sivaranjani M.G, Swetha JK
Sprint-2	Prediction	USN-2	Cleaning the dataset for processing	10	High	Sivaranjani M.G, Swetha S
Sprint-2	Prediction	USN-1	Process the data and choosing the effective algorithm	10	High	Swetha S, Swetha JK
Sprint-2	Sign Up	USN-1	Enabling G-mail sign up	10	High	Srinidhi R
Sprint-4	Flask Integration	USN-1	Integrating flask with the model	10	High	Swetha S

6.2 Sprint Delivery Schedule:

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2022	30 Oct 2022	20	30 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	05 Nov 2022
Sprint-3	20	6 Days	06 Nov 2022	11 Nov 2022	20	11 Nov 2022
Sprint-4	20	6 Days	12 Nov 2022	17 Nov 2022	20	17 Nov 2022

6.3 REPORTS FROM JIRA REPORT

Issue Type	Key	Summary	Assignee	Reporter	Priority	Status	Resolution	Created	Updated	Due date
Story	CKD-5	As a user, I can log into the application by entering email & password	Sowmiya E	SWETHA S	High	Done	Done	01-11-2022 13:01	01-11-2022 13:29	
Story	CKD-4	As a user, I can log into the application by entering email & password	sivaranjani	SWETHA S	High	Done	Done	01-11-2022 12:58	01-11-2022 13:45	
Story	CKD-3	As a user, I will receive confirmation email once I have registered for the	SWETHA S	SWETHA S	High	Done	Done	01-11-2022 12:58	01-11-2022 13:29	
Story	CKD-2	As a user, I can change my password.	srinidhi.r.2	SWETHA S	High	Done	Done	01-11-2022 12:57	01-11-2022 13:29	
Story	CKD-1	As a user, I can register for the application by entering my email, passwo	swetha.jk	SWETHA S	High	Done	Done	01-11-2022 12:57	01-11-2022 13:53	

Issue Type	Key	Summary	Assignee	Reporter	Priority	Status	Resolution	Created	Updated	Due date
Story	CKD-10	Choosing t	swetha.jk	SWETHA S	Medium	Done	Done	01-11-2022 13:08	14-11-2022 11:02	
Story	CKD-9	As a user, I	srinidhi.r.2	SWETHA S	Medium	Done	Done	01-11-2022 13:06	14-11-2022 11:02	
Story	CKD-8	As a user, I	Sowmiya E	SWETHA S	Medium	Done	Done	01-11-2022 13:06	14-11-2022 11:02	
Story	CKD-7	Choosing t	SWETHA S	SWETHA S	Medium	Done	Done	01-11-2022 13:05	14-11-2022 11:02	
Story	CKD-6	Clean the (	sivaranjani	SWETHA S	Medium	Done	Done	01-11-2022 13:05	14-11-2022 11:01	

Issue Type	Key	Summary	Assignee	Reporter	Priority	Status	Resolution	Created	Updated	Due date
Story	CKD-15	As a user, I	swetha.jk	SWETHA S	Medium	Done	Done	01-11-2022 13:15	16-11-2022 17:30	
Story	CKD-14	As a user, I	Sowmiya E	SWETHA S	Medium	Done	Done	01-11-2022 13:14	18-11-2022 11:07	
Story	CKD-13	As a user, I	srinidhi.r.2	SWETHA S	Medium	Done	Done	01-11-2022 13:14	18-11-2022 11:07	
Story	CKD-12	As a user, I	sivaranjani	SWETHA S	Medium	Done	Done	01-11-2022 13:14	16-11-2022 17:29	
Story	CKD-11	As a user, I	SWETHA S	SWETHA S	Medium	Done	Done	01-11-2022 13:11	18-11-2022 11:07	

Issue Type	Key	Summary	Assignee	Reporter	Priority	Status	Resolution	Created	Updated
Story	CKD-20	As a user, I	swetha.jk	SWETHA S	Medium	Done	Done	01-11-2022 13:16	18-11-2022 11:07
Story	CKD-19	As a user, I	srinidhi.r.2	SWETHA S	Medium	Done	Done	01-11-2022 13:16	18-11-2022 11:07
Story	CKD-18	As a user, I	Sowmiya E	SWETHA S	Medium	Done	Done	01-11-2022 13:16	18-11-2022 11:07
Story	CKD-17	As a user, I	sivaranjani	SWETHA S	Medium	Done	Done	01-11-2022 13:16	18-11-2022 11:07
Story	CKD-16	As a user, I	SWETHA S	SWETHA S	Medium	Done	Done	01-11-2022 13:16	18-11-2022 11:07

7.CODING & SOLUTIONING

7.1 Feature 1:

AUTHENTICATION:

The user can register by providing the details needed. Then, the user gets a confirmation mail. A token that is valid for a specified time limit is given for verification. After verification, the user can login and proceed with the next steps to get the prediction result.

```
@app.route("/Register", methods=['GET', 'POST'])
def register():
    if current_user.is_authenticated:
        return redirect(url_for('home'))
    form = RegistrationForm()
    if form.validate_on_submit():
```

```
        hashed_password =
bcrypt.generate_password_hash(form.password.data).decode('utf-8')
        user = User(username=form.username.data, email=form.email.data,
password=hashed_password)
        db.session.add(user)
        db.session.commit()
        flash('Your account has been created! You are now able to log in',
'success')
        return redirect(url_for('login'))
    return render_template('Register.html', title='Register', form=form)

'''
def send_conf_email(user):
    token = user.get_reset_token()
    msg = Message('Confirmation Mail',
                  sender='kidney.disease.predictor@gmail.com',
                  recipients=[user.email])
    msg.body = f'''Your account was successfully created. Please click the
link below to confirm your email address and activate your account:

{url_for('register_token', token=token, _external=True)}

If you did not make this request then simply ignore this email and no changes
will be made.
'''
    mail.send(msg)

@app.route("/Register", methods=['GET', 'POST'])
def register():
    if current_user.is_authenticated:
        return redirect(url_for('home'))
    form = RegistrationForm()
    if form.validate_on_submit():
        hashed_password =
bcrypt.generate_password_hash(form.password.data).decode('utf-8')
        user = User(username=form.username.data, email=form.email.data,
password=hashed_password)
        db.session.add(user)
        db.session.commit()
        user = User.query.filter_by(email=form.email.data).first()
        send_conf_email(user)
        flash('An email has been sent with instructions to confirm your
account.', 'info')
        return redirect(url_for('login'))
    return render_template('Register.html', title='Register', form=form)

@app.route("/Register/<token>", methods=['GET', 'POST'])
def register_token(token):
```

```
    if current_user.is_authenticated:
        return redirect(url_for('home'))
    user = User.verify_reset_token(token)
    if user is None:
        flash('That is an invalid or expired token', 'warning')
        return redirect(url_for('register'))
    user.email_confirmed = True
    db.session.commit()
    flash('Your email is verified! You are now able to log in', 'success')
    return redirect(url_for('login'))

@app.route("/login", methods=['GET', 'POST'])
def login():
    if current_user.is_authenticated:
        return redirect(url_for('home'))
    form = LoginForm()
    if form.validate_on_submit():
        user = User.query.filter_by(email=form.email.data).first()
        if user and bcrypt.check_password_hash(user.password,
form.password.data):
            login_user(user, remember=form.remember.data)
            next_page = request.args.get('next')
            return redirect(next_page) if next_page else
redirect(url_for('home'))
        else:
            flash('Login Unsuccessful. Please check email and password',
'danger')
    return render_template('login.html', title='Login', form=form)


@app.route("/logout")
def logout():
    logout_user()
    return redirect(url_for('home'))


@app.route("/account")
@login_required
def account():
    return render_template('account.html', title='Account')

@app.route('/predict', methods=['GET'])
@login_required
def predict():
    return render_template('prediction.html')
```

7.2 Feature 2:

The user can also reset the password, if needed. If the user has forgotten the password, it can be reset by clicking "**Forgot Password**" and getting a verification mail to reset the password.

```
def send_reset_email(user):
    token = user.get_reset_token()
    msg = Message('Password Reset Request',
                  sender='kidney.disease.predictor@gmail.com',
                  recipients=[user.email])
    msg.body = f'''To reset your password, visit the following link:
{url_for('reset_token', token=token, _external=True)}

If you did not make this request then simply ignore this email and no changes
will be made.
'''
    mail.send(msg)

@app.route('/reset_password', methods=['GET','POST'])
def reset_request():
    if current_user.is_authenticated:
        return redirect(url_for('home'))
    form = RequestResetForm()
    if form.validate_on_submit():
        user = User.query.filter_by(email=form.email.data).first()
        send_reset_email(user)
        flash('An email has been sent with instructions to reset your
password.', 'info')
        return redirect(url_for('login'))
    return render_template('forgot_password.html', form=form)

@app.route("/reset_password/<token>", methods=['GET', 'POST'])
def reset_token(token):
    if current_user.is_authenticated:
        return redirect(url_for('home'))
    user = User.verify_reset_token(token)
    if user is None:
        flash('That is an invalid or expired token', 'warning')
        return redirect(url_for('reset_request'))
    form = ResetPasswordForm()
    if form.validate_on_submit():
        hashed_password =
bcrypt.generate_password_hash(form.password.data).decode('utf-8')
        user.password = hashed_password
        db.session.commit()
        flash('Your password has been updated! You are now able to log in',
'success')
        return redirect(url_for('login'))
```

```
    return render_template('change_password.html', title='Reset Password',
form=form)
```

7.3 FEATURE 3:

G-MAIL LOGIN:

The user has to login every time to view the prediction page. As it displays the health condition of the user, it needs to be secure such that unauthorized users will not be able to access it.

The user need not enter the login credentials every time to visit the page. Instead, the user can login using the G-mail account. Thus, the app provides login facilities with security**.**

```
@app.route("/glogin")
def glogin():
    google_provider_cfg = get_google_provider_cfg()
    authorization_endpoint = google_provider_cfg["authorization_endpoint"]
    request_uri = client.prepare_request_uri(
        authorization_endpoint,
        redirect_uri=request.base_url + "/callback",

        scope=["openid", "email", "profile"],
    )
    return redirect(request_uri)


@app.route("/glogin/callback")
def callback():

    code = request.args.get("code")
    google_provider_cfg = get_google_provider_cfg()
    token_endpoint = google_provider_cfg["token_endpoint"]


    token_url, headers, body = client.prepare_token_request(
        token_endpoint,
        authorization_response=request.url,
        redirect_url=request.base_url,
```

```
        code=code,
    )
    token_response = requests.post(
        token_url,
        headers=headers,
        data=body,
        auth=(GOOGLE_CLIENT_ID, GOOGLE_CLIENT_SECRET),
    )

    client.parse_request_body_response(json.dumps(token_response.json()))

    userinfo_endpoint = google_provider_cfg["userinfo_endpoint"]
    uri, headers, body = client.add_token(userinfo_endpoint)
    userinfo_response = requests.get(uri, headers=headers, data=body)

    if userinfo_response.json().get("email_verified"):
        users_email = userinfo_response.json()["email"]
        users_name = userinfo_response.json()["name"]
    else:
        return "User email not available or not verified by Google.", 400
    password =  bcrypt.generate_password_hash(users_name).decode('utf-8')
    user = User(
        username=users_name, email=users_email,email_confirmed=True,password =
password
    )
    db.session.add(user)
    db.session.commit()
    login_user(user)

    return redirect(url_for("home"))

def get_google_provider_cfg():
    return requests.get(GOOGLE_DISCOVERY_URL).json()
```

7.4 Database Schema:

In this model, when the user tries to log in, it is checked whether the user has an account. If not, the user is directed to the ‘Registration Page’. The user can register and then login.

There are five fields: Username, password, E-mail,E-mail confirmed, ID.

```
class User(db.Model, UserMixin):
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(20), unique=True, nullable=False)
    email = db.Column(db.String(120), unique=True, nullable=False)
    email_confirmed = db.Column(db.Boolean, default=False, nullable=False)
    password = db.Column(db.String(60), nullable=False)

    def get_reset_token(self, expires_sec=1800):
        s = Serializer(app.config['SECRET_KEY'], expires_sec)
        return s.dumps({'user_id': self.id}).decode('utf-8')

    @staticmethod
    def verify_reset_token(token):
        s = Serializer(app.config['SECRET_KEY'])
        try:

           user_id = s.loads(token)['user_id']
        except:
            return None
        return User.query.get(user_id)
```

RESULTS

Performance Metrics:

- The performance of the model is measured by its accuracy level. The higher the accuracy level, the more reliable the application is to the users.

- The application has features that are user-friendly and is secure, therefore providing satisfaction to the users.

	0	1
0	LogisticRegression	0.9875
1	Supportvectormachines	0.9625
2	GaussianNB	1.0000
3	MultinomialNB	0.8750
4	SGDClassifier	0.3500
5	KNeighborsClassifier	0.6875
6	DecisionTreeClassifier	0.9875
7	RandomForestClassifier	1.0000
8	GradientBoostingClassifier	1.0000
9	LGBMClassifier	1.0000
10	XGBClassifier	0.9875
11	AdaBoostClassifier	1.0000

ADVANTAGES & DISADVANTAGES:

Advantages:

1. **Reliability**

The prediction is accurate. Thus, the users can find the app to be reliable.

2.Better Understanding

As the results are accurate, the users can gain clarity and a better understanding on their kidneys' condition that would help them to decide what to do next.

3. Develops Awareness among Users

Apart from deciding on what to do next, the users also develop the awareness to get regular health check-ups. This helps them to keep track of their health condition and stay updated.

4. Effective Treatment

If CKD is detected earlier, it can be treated to slow down its progression. Hence, by delaying the progression the severity of CKD can also be delayed or halted.

5. Staying healthy

By getting tested regularly, the patients who are diagnosed with CKD can keep track of their health condition and take the necessary steps like following customized diet plans and staying hydrated to improve their health.

Disadvantages:

1. As the prediction results are based on the data provided by the users, it is necessary that the user is extremely careful with the values they are entering. If the values are inappropriate, it might impact the prediction results.

CONCLUSION

The goal of the project was to predict the presence chronic kidney disease of a person based on the data given. The algorithm used in this model is the Random Forest Algorithm. This algorithm proves to be highly accurate. The features that are of utmost importance are considered to detect CKD. The least important features are eliminated. This feature reduction is done by Recursive Feature Elimination method. This is done so that the prediction is more accurate. Thus, the goal is achieved.

The user can register, login and then enter the required data to get their prediction result. If the user has forgotten the password or wants to reset it, this can also be done as the app provides the necessary facility to do so. The user can also login using Google account. These features make the app user-friendly.

The app makes accurate predictions regarding the presence of CKD. This makes it more reliable. The users can get to know their kidneys' condition and develop awareness about kidney health to be more cautious.

Even though the results are accurate, it depends wholly on the data given by the users. Thus, if the data entered by the user is inappropriate, it might affect the prediction result.

FUTUTRE SCOPE

As CKD has become an increasing issue in the present world, people are becoming more cautious and taking steps to get a better understating of their health condition.
In future, such early prediction systems will become more popular as the present world demands such systems.
Earlier prediction of CKD will help the patients to get effective treatment as the treatment for advanced stages are more expensive and difficult.

APPENDIX

SOURCE CODE:

```
#__init__.py

import requests
from flask_mail import Mail
from flask_bcrypt import Bcrypt
from flask_login import UserMixin
from flask_sqlalchemy import SQLAlchemy
from flask import Flask,request,redirect,url_for
from itsdangerous import TimedJSONWebSignatureSerializer as Serializer
from flask_login import login_user, current_user, logout_user,
login_required,LoginManager

API_KEY = "_YBwCIVTCKowMStvz4HHQTcizrYGJUV0p9bj_W4uzh3-"
token_response = requests.post('https://iam.cloud.ibm.com/identity/token',
data={"apikey":
API_KEY, "grant_type": 'urn:ibm:params:oauth:grant-type:apikey'})
mltoken = token_response.json()["access_token"]
header = {'Content-Type': 'application/json', 'Authorization': 'Bearer ' +
mltoken}
import os
os.environ['OAUTHLIB_INSECURE_TRANSPORT'] = '1'
app = Flask(__name__)
app.debug = True
app.config['SECRET_KEY'] = '5791628bb0b13ce0c676dfde280ba245'
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///site.db'

bcrypt = Bcrypt(app)
login_manager = LoginManager(app)

app.config['MAIL_SERVER'] = 'smtp.googlemail.com'
app.config['MAIL_PORT'] = 587
```

```
app.config['MAIL_USE_TLS'] = True
app.config['MAIL_USERNAME'] ="kidney.disease.predictor@gmail.com"
app.config['MAIL_PASSWORD'] = "gfufjkvasogorlmr"
mail = Mail(app)
login_manager.login_view = 'login'
login_manager.login_message_category = 'info'

@login_manager.user_loader
def load_user(user_id):
    return User.query.get(int(user_id))
db = SQLAlchemy(app)

class User(db.Model, UserMixin):
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(20), unique=True, nullable=False)
    email = db.Column(db.String(120), unique=True, nullable=False)
    email_confirmed = db.Column(db.Boolean, default=False, nullable=False)
    password = db.Column(db.String(60), nullable=False)

    def get_reset_token(self, expires_sec=1800):
        s = Serializer(app.config['SECRET_KEY'], expires_sec)
        return s.dumps({'user_id': self.id}).decode('utf-8')

    @staticmethod
    def verify_reset_token(token):
        s = Serializer(app.config['SECRET_KEY'])
        try:

           user_id = s.loads(token)['user_id']
        except:
            return None
        return User.query.get(user_id)
from chronic_pred import routes
```

```
#forms.py
from flask_wtf import FlaskForm
from wtforms import StringField, PasswordField, SubmitField, BooleanField
from wtforms.validators import DataRequired, Length, Email, EqualTo,
ValidationError
from chronic_pred import User


class RegistrationForm(FlaskForm):
    username = StringField('Username',
                           validators=[DataRequired(), Length(min=2, max=20)])
    email = StringField('Email',
                        validators=[DataRequired(), Email()])
    password = PasswordField('Password', validators=[DataRequired()])
```

```
    confirm_password = PasswordField('Confirm Password',
                                     validators=[DataRequired(),
EqualTo('password')])
    submit = SubmitField('Sign Up')

    def validate_username(self, username):
        user = User.query.filter_by(username=username.data).first()
        if user:
            raise ValidationError('That username is taken. Please choose a
different one.')

    def validate_email(self, email):
        user = User.query.filter_by(email=email.data).first()
        if user:
            raise ValidationError('That email is taken. Please choose a
different one.')


class LoginForm(FlaskForm):
    email = StringField('Email',
                        validators=[DataRequired(), Email()])
    password = PasswordField('Password', validators=[DataRequired()])
    remember = BooleanField('Remember Me')
    submit = SubmitField('Login')

class RequestResetForm(FlaskForm):
    email = StringField('Email',
                        validators=[DataRequired(), Email()])
    submit = SubmitField('Request Password Reset')

    def validate_email(self, email):
        user = User.query.filter_by(email=email.data).first()
        if user is None:
            raise ValidationError('There is no account with that email. You
must register first.')

class ResetPasswordForm(FlaskForm):
    password = PasswordField('Password', validators=[DataRequired()])
    confirm_password = PasswordField('Confirm Password',
                                     validators=[DataRequired(),
EqualTo('password')])
    submit = SubmitField('Reset Password')
```

```
#routes.py
import json
import requests
from chronic_pred import User
```

```
from flask_mail import Message
from oauthlib.oauth2 import WebApplicationClient
from chronic_pred import app, db, bcrypt, mail
from chronic_pred import mltoken
from flask import render_template, url_for, flash, redirect, request
from chronic_pred.forms import RegistrationForm, LoginForm, ResetPasswordForm,
RequestResetForm
from flask_login import login_user, current_user, logout_user, login_required


GOOGLE_CLIENT_ID = "660143345755-
c53v3c2qe7ejoj0r0vc9op5am8hksv82.apps.googleusercontent.com"
GOOGLE_CLIENT_SECRET = "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"
GOOGLE_DISCOVERY_URL = (
    "https://accounts.google.com/.well-known/openid-configuration"
)

client = WebApplicationClient(GOOGLE_CLIENT_ID)

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

@app.route("/links")
@login_required
def useful_links():
    return render_template('blog.html')

'''
@app.route("/Register", methods=['GET', 'POST'])
def register():
    if current_user.is_authenticated:
        return redirect(url_for('home'))
    form = RegistrationForm()
    if form.validate_on_submit():
        hashed_password =
bcrypt.generate_password_hash(form.password.data).decode('utf-8')
        user = User(username=form.username.data, email=form.email.data,
password=hashed_password)
        db.session.add(user)
        db.session.commit()
        flash('Your account has been created! You are now able to log in',
'success')
        return redirect(url_for('login'))
    return render_template('Register.html', title='Register', form=form)

'''
def send_conf_email(user):
```

```
    token = user.get_reset_token()
    msg = Message('Confirmation Mail',
                  sender='kidney.disease.predictor@gmail.com',
                  recipients=[user.email])
    msg.body = f'''Your account was successfully created. Please click the
link below to confirm your email address and activate your account:

{url_for('register_token', token=token, _external=True)}

If you did not make this request then simply ignore this email and no changes
will be made.
'''
    mail.send(msg)

@app.route("/Register", methods=['GET', 'POST'])
def register():
    if current_user.is_authenticated:
        return redirect(url_for('home'))
    form = RegistrationForm()
    if form.validate_on_submit():
        hashed_password =
bcrypt.generate_password_hash(form.password.data).decode('utf-8')
        user = User(username=form.username.data, email=form.email.data,
password=hashed_password)
        db.session.add(user)
        db.session.commit()
        user = User.query.filter_by(email=form.email.data).first()
        send_conf_email(user)
        flash('An email has been sent with instructions to confirm your
account.', 'info')
        return redirect(url_for('login'))
    return render_template('Register.html', title='Register', form=form)

@app.route("/Register/<token>", methods=['GET', 'POST'])
def register_token(token):
    if current_user.is_authenticated:
        return redirect(url_for('home'))
    user = User.verify_reset_token(token)
    if user is None:
        flash('That is an invalid or expired token', 'warning')
        return redirect(url_for('register'))
    user.email_confirmed = True
    db.session.commit()
    flash('Your email is verified! You are now able to log in', 'success')
    return redirect(url_for('login'))

@app.route("/login", methods=['GET', 'POST'])
def login():
```

```
    if current_user.is_authenticated:
        return redirect(url_for('home'))
    form = LoginForm()
    if form.validate_on_submit():
        user = User.query.filter_by(email=form.email.data).first()
        if user and bcrypt.check_password_hash(user.password,
form.password.data):
            login_user(user, remember=form.remember.data)
            next_page = request.args.get('next')
            return redirect(next_page) if next_page else
redirect(url_for('home'))
        else:
            flash('Login Unsuccessful. Please check email and password',
'danger')
    return render_template('login.html', title='Login', form=form)


@app.route("/logout")
def logout():
    logout_user()
    return redirect(url_for('home'))


@app.route("/account")
@login_required
def account():
    return render_template('account.html', title='Account')

@app.route('/predict', methods=['GET'])
@login_required
def predict():
    return render_template('prediction.html')

def send_reset_email(user):
    token = user.get_reset_token()
    msg = Message('Password Reset Request',
                  sender='kidney.disease.predictor@gmail.com',
                  recipients=[user.email])
    msg.body = f'''To reset your password, visit the following link:
{url_for('reset_token', token=token, _external=True)}

If you did not make this request then simply ignore this email and no changes
will be made.
'''
    mail.send(msg)

@app.route('/reset_password', methods=['GET','POST'])
def reset_request():
    if current_user.is_authenticated:
```

```
        return redirect(url_for('home'))
    form = RequestResetForm()
    if form.validate_on_submit():
        user = User.query.filter_by(email=form.email.data).first()
        send_reset_email(user)
        flash('An email has been sent with instructions to reset your
password.', 'info')
        return redirect(url_for('login'))
    return render_template('forgot_password.html', form=form)

@app.route("/reset_password/<token>", methods=['GET', 'POST'])
def reset_token(token):
    if current_user.is_authenticated:
        return redirect(url_for('home'))
    user = User.verify_reset_token(token)
    if user is None:
        flash('That is an invalid or expired token', 'warning')
        return redirect(url_for('reset_request'))
    form = ResetPasswordForm()
    if form.validate_on_submit():
        hashed_password =
bcrypt.generate_password_hash(form.password.data).decode('utf-8')
        user.password = hashed_password
        db.session.commit()
        flash('Your password has been updated! You are now able to log in',
'success')
        return redirect(url_for('login'))
    return render_template('change_password.html', title='Reset Password',
form=form)

@app.route('/result', methods=['GET','POST'])
def result():
    if request.method == 'POST':
        age = int(request.form['age'])
        usg = float(request.form['usg'])
        sal = float(request.form['sal'])
        bp = float(request.form['bp'])
        su = float(request.form['su'])
        bu = float(request.form['bu'])
        sc = float(request.form['sc'])
        sod = float(request.form['sod'])
        pot = float(request.form['pot'])
        hg = float(request.form['hg'])
        pcv = int(request.form['pcv'])
        wbcc = float(request.form['wbcc'])
        diab = str(request.form['diab'])
        diab = 1 if diab == 'yes' else 0
        cad = str(request.form['cad'])
```

```
        cad = 1 if cad == 'yes' else 0
        hp = str(request.form['hp'])
        hp = 1 if hp == 'yes' else 0
        X = [[age, usg, sal, bp, su, bu, sc, sod, pot, hg, pcv, wbcc, diab,
cad, hp]]
        payload_scoring = {
        "input_data": [{"field": [["bp", "usg", "sal"," su", "bu", "sc",
"sod", "pot", "hg", "pcv", "wbcc", "diab", "cad", "hp" ,"age"]], "values":
[[bp, usg, sal, su, bu, sc, sod, pot, hg, pcv, wbcc, diab, cad, hp ,age]]}]
}
        response_scoring = requests.post('https://us-
south.ml.cloud.ibm.com/ml/v4/deployments/c987d410-9ac4-4691-9ddf-
cbb42f7b4516/predictions?version=2022-11-13', json=payload_scoring,
headers={'Authorization': 'Bearer ' + mltoken})
        #print("Scoring response")
        predictions = response_scoring.json()
        print("Scoring response")
        print(response_scoring.json())
        predict = predictions["predictions"][0]["values"][0][0]
        if predict == 1:
            return render_template('negative.html')
        else:
            return render_template('positive.html')

@app.route("/glogin")
def glogin():
    google_provider_cfg = get_google_provider_cfg()
    authorization_endpoint = google_provider_cfg["authorization_endpoint"]
    request_uri = client.prepare_request_uri(
        authorization_endpoint,
        redirect_uri=request.base_url + "/callback",
        scope=["openid", "email", "profile"],
    )
    return redirect(request_uri)


@app.route("/glogin/callback")
def callback():

    code = request.args.get("code")
    google_provider_cfg = get_google_provider_cfg()
    token_endpoint = google_provider_cfg["token_endpoint"]


    token_url, headers, body = client.prepare_token_request(
        token_endpoint,
        authorization_response=request.url,
        redirect_url=request.base_url,
        code=code,
```

```
    )
    token_response = requests.post(
        token_url,
        headers=headers,
        data=body,
        auth=(GOOGLE_CLIENT_ID, GOOGLE_CLIENT_SECRET),
    )

    client.parse_request_body_response(json.dumps(token_response.json()))

    userinfo_endpoint = google_provider_cfg["userinfo_endpoint"]
    uri, headers, body = client.add_token(userinfo_endpoint)
    userinfo_response = requests.get(uri, headers=headers, data=body)

    if userinfo_response.json().get("email_verified"):
        users_email = userinfo_response.json()["email"]
        users_name = userinfo_response.json()["name"]
    else:
        return "User email not available or not verified by Google.", 400
    password =  bcrypt.generate_password_hash(users_name).decode('utf-8')
    user = User(
        username=users_name, email=users_email,email_confirmed=True,password =
password
    )
    db.session.add(user)
    db.session.commit()
    login_user(user)

    return redirect(url_for("home"))

def get_google_provider_cfg():
    return requests.get(GOOGLE_DISCOVERY_URL).json()
```

```
#run.py

 from chronic_pred import app

if __name__ == '__main__':

    app.run(debug=True)
```

```
Templates
```

index.html

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="utf-8">
  <meta content="width=device-width, initial-scale=1.0" name="viewport">

  <title>Chronic kidney disease predictor</title>
  <meta content="" name="description">
  <meta content="" name="keywords">

  <!-- Google Fonts -->
  <link
href="https://fonts.googleapis.com/css?family=Open+Sans:300,300i,400,400i,600,
600i,700,700i|Raleway:300,300i,400,400i,500,500i,600,600i,700,700i|Poppins:300
,300i,400,400i,500,500i,600,600i,700,700i" rel="stylesheet">

  <!-- Vendor CSS Files -->

  <link rel="stylesheet" href="{{ url_for('static', filename='aos.css') }}">
  <link rel="stylesheet" href="{{ url_for('static',
filename='bootstrap.min.css') }}">
  <link rel="stylesheet" href="{{ url_for('static', filename='bootstrap-
icons.css') }}">
  <link rel="stylesheet" href="{{ url_for('static',
filename='boxicons.min.css') }}">
  <link rel="stylesheet" href="{{ url_for('static',
filename='glightbox.min.css') }}">
  <link rel="stylesheet" href="{{ url_for('static', filename='remixicon.css')
}}">
  <link rel="stylesheet" href="{{ url_for('static', filename='swiper-
bundle.min.css') }}">

  <!-- Template Main CSS File -->
  <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">

</head>

<body style="background-image: url('static/hero-bg.jpg');">

  <!-- ======= Header ======= -->
  <header id="header" class="fixed-top">
    <div class="container d-flex align-items-center justify-content-between">
      <h1 class="logo"><a href="index.html"></a>preventCKD</h1>

      <nav id="navbar" class="navbar">
```

```
        <ul>
          {% if current_user.is_authenticated %}
          <li><a href="{{ url_for('useful_links') }}">Useful links</a></li>
          <li><a href="{{ url_for('predict') }}"> Predict </a></li>
          <li><a href="{{ url_for('logout') }}">Logout</a></li>

          {% else %}
          <li><a href="{{ url_for('login') }}"> Sign in </a></li>
          <li><a class="getstarted scrollto" href="{{ url_for('register')
}}">Get Started</a></li>
          {% endif %}
        </ul>
        <i class="bi bi-list mobile-nav-toggle"></i>
      </nav><!-- .navbar -->

    </div>
  </header><!-- End Header -->
  <div>{% with messages = get_flashed_messages(with_categories=true) %}
    {% if messages %}
      {% for category, message in messages %}
        <div class="alert alert-{{ category }}">
          {{ message }}
        </div>
      {% endfor %}
    {% endif %}
  {% endwith %}
  {% block content %}{% endblock %}</div>
  <!-- ======= Hero Section ======= -->
  <section id="hero" class="d-flex align-items-center">
    <div class="container position-relative" data-aos="fade-up" data-aos-
delay="100">
      <div class="row justify-content-center">
        <div class="col-xl-7 col-lg-9 text-center">
          <h1>Your Kidney has an important job to do</h1>
          <h2>We help you to keep your kidney healthy</h2>
        </div>
      </div>
      <div class="text-center">
        <a href="{{ url_for('register') }}" class="btn-get-started
scrollto">Get Started</a>

      </div>

      <div class="row icon-boxes">
        <div class="col-md-6 col-lg-3 d-flex align-items-stretch mb-5 mb-lg-0"
data-aos="zoom-in" data-aos-delay="200">
          <div class="icon-box">
```

```
            <h4 class="title"><a href="">Your Kidneys Pump More than 50
Gallons of Blood Daily</a></h4>
            <p class="description">Your kidneys may be small, but they do
quite a bit of work! The kidneys are responsible for removing waste and excess
liquid from the body by filtering them out from your blood. Your kidneys
filter through about 52 gallons (200 liters) of blood throughout a given
day!</p>
          </div>
        </div>

        <div class="col-md-6 col-lg-3 d-flex align-items-stretch mb-5 mb-lg-0"
data-aos="zoom-in" data-aos-delay="300">
          <div class="icon-box">
            <h4 class="title"><a href="">They Regulate Your Body’s Salt
Content</a></h4>
            <p class="description">Along with filtering out waste from your
blood, your kidneys also help regulate your body’s sodium levels. However, it
is important to keep in mind that it is possible to take in more salt than
your kidneys can safely remove from your body. While salt is essential for
your body to function properly, excess amounts can be damaging to your body,
leading to heart disease, stroke, and even kidney failure.</p>
          </div>
        </div>

        <div class="col-md-6 col-lg-3 d-flex align-items-stretch mb-5 mb-lg-0"
data-aos="zoom-in" data-aos-delay="400">
          <div class="icon-box">
            <h4 class="title"><a href="">Kidney Disease, a huge
burden</a></h4>
            <p class="description"> Kidney disease remains a huge burden on
the society and it is estimated that about 10% of the adult population has
some form of kidney disease and 200,000 people get afflicted with severe
kidney disease (end stage kidney disease) every year </p>
          </div>
        </div>

        <div class="col-md-6 col-lg-3 d-flex align-items-stretch mb-5 mb-lg-0"
data-aos="zoom-in" data-aos-delay="500">
          <div class="icon-box">
            <h4 class="title"><a href="">kidney disease, preventable?</a></h4>
            <p class="description">Many kidney diseases are preventable if
detected early and certainly the progression of kidney disease can be slowed
down with good control of diabetes, hypertension, lifestyle modifications,
dietary measures and avoidance of medicines that are toxic to the kidneys</p>
          </div>
        </div>

      </div>
```

```
    </div>
  </section><!-- End Hero -->


  <div id="preloader"></div>
  <a href="#" class="back-to-top d-flex align-items-center justify-content-
center"><i class="bi bi-arrow-up-short"></i></a>

  <!-- Vendor JS Files -->

  <script src="{{ url_for('static', filename='purecounter_vanilla.js')
}}"></script>
  <script src="{{ url_for('static', filename='aos.js') }}"></script>
  <script src="{{ url_for('static', filename='bootstrap.bundle.min.js')
}}"></script>
  <script src="{{ url_for('static', filename='glightbox.min.js') }}"></script>
  <script src="{{ url_for('static', filename='isotope.pkgd.min.js')
}}"></script>
  <script src="{{ url_for('static', filename='swiper-bundle.min.js')
}}"></script>
  <script src="{{ url_for('static', filename='validate.js') }}"></script>

  <!-- Template Main JS File -->
  <script src="{{ url_for('static', filename='main.js') }}"></script>

</body>

</html>
```

```
Login.html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Login</title>

    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
1BmE4kWBq78iYhFldvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoqyl2QvZ6jIW3"
crossorigin="anonymous"/>
    <link rel="stylesheet"
href="https://use.fontawesome.com/releases/v5.0.8/css/all.css">
    <style>
    .form-control form-control-lg{
      font-size: 18px;
```

```
    }
    .btn-primary {
    color: #fff;
    background-color: #2487ce;
    border-color: #2487ce;
}
</style>

  </head>
  <body style="font-family: 'Times New Roman', Times, serif;">
    <section>
      <div class="row" style="width:100%"><div class="col-md-12">
        {% with messages = get_flashed_messages(with_categories=true) %}
          {% if messages %}
            {% for category, message in messages %}
              <div class="alert alert-{{ category }}">
                {{ message }}
              </div>
            {% endfor %}
          {% endif %}
        {% endwith %}
        {% block content %}{% endblock %}
      </div></div>


      <div
        class="d-flex flex-column min-vh-100 justify-content-center">
        <div class="container">
            <div class="container-fluid h-custom">
              <div class="row d-flex justify-content-center align-items-center
h-100">
                <div class="col-md-9 col-lg-6 col-xl-5">
                  <img src="https://mdbcdn.b-cdn.net/img/Photos/new-
templates/bootstrap-login-form/draw2.webp"
                    class="img-fluid" alt="Sample image">
                </div>
                <div class="col-md-8 col-lg-6 col-xl-4 offset-xl-1">
                  <form method="POST" action="">
                    {{ form.hidden_tag() }}
                    <fieldset class="form-group">
                        <legend class="border-bottom mb-4">Log In</legend>
                        <div class="form-group">
                            {{ form.email.label(class="form-control-label") }}
                            {% if form.email.errors %}
                                {{ form.email(class="form-control form-
control-lg is-invalid") }}
                                <div class="invalid-feedback">
                                    {% for error in form.email.errors %}
```

```
                                    <span>{{ error }}</span>
                                {% endfor %}
                            </div>
                        {% else %}
                            {{ form.email(class="form-control form-
control-lg") }}
                        {% endif %}
                    </div>
                    <div class="form-group">
                        {{ form.password.label(class="form-control-label")
}}
                        {% if form.password.errors %}
                            {{ form.password(class="form-control form-
control-lg is-invalid") }}
                            <div class="invalid-feedback">
                                {% for error in form.password.errors %}
                                    <span>{{ error }}</span>
                                {% endfor %}
                            </div>
                        {% else %}
                            {{ form.password(class="form-control form-
control-lg") }}
                        {% endif %}
                    </div><br>
                    <div class="form-check">
                        {{ form.remember(class="form-check-input") }}
                        {{ form.remember.label(class="form-check-label")
}}
                    </div>
                    <br>
                </fieldset>
                <div class="form-group">
                    {{ form.submit(class="btn btn-primary") }}
                </div><br>
                <small class="text-muted ml-2">
                    <a href="{{ url_for('reset_request') }}">Forgot
Password?</a>
                </small>
            </form>
        </div>
    </div>
</div>
</div>
</section>
</body>
</html>
```

Register.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Get started</title>

    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
1BmE4kWBq78iYhFldvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoqyl2QvZ6jIW3"
crossorigin="anonymous"/>
    <link rel="stylesheet"
href="https://use.fontawesome.com/releases/v5.0.8/css/all.css">
    <style>
    .form-control form-control-lg{
      font-size: 18px;
    }
    .btn-primary {
    color: #fff;
    background-color: #2487ce;
    border-color: #2487ce;
}
</style>

  </head>
  <body style="font-family: 'Times New Roman', Times, serif;" >
    <section>
      <div class="row" style="width:100%"><div class="col-md-12">
        {% with messages = get_flashed_messages(with_categories=true) %}
          {% if messages %}
            {% for category, message in messages %}
              <div class="alert alert-{{ category }}">
                {{ message }}
              </div>
            {% endfor %}
          {% endif %}
        {% endwith %}
        {% block content %}{% endblock %}
      </div></div>
      <div
        class="d-flex flex-column min-vh-100 justify-content-center">
        <div class="container">
            <div class="container-fluid h-custom">
              <div class="row d-flex justify-content-center align-items-center
h-100">
```

```
                <div class="col-md-9 col-lg-6 col-xl-5">
                  <img src="https://mdbcdn.b-cdn.net/img/Photos/new-
templates/bootstrap-login-form/draw2.webp"
                    class="img-fluid" alt="Sample image">
                </div>
                <div class="col-md-8 col-lg-6 col-xl-4 offset-xl-1">
                  <form method="POST" action="">
                    {{ form.hidden_tag() }}
                    <fieldset class="form-group">
                        <legend class="border-bottom mb-4">Join Today</legend>
                        <div class="form-group">
                            {{ form.username.label(class="form-control-label")
}}

                            {% if form.username.errors %}
                                {{ form.username(class="form-control form-
control-lg is-invalid") }}
                                <div class="invalid-feedback">
                                    {% for error in form.username.errors %}
                                        <span>{{ error }}</span>
                                    {% endfor %}
                                </div>
                            {% else %}
                                {{ form.username(class="form-control form-
control-lg") }}
                            {% endif %}
                        </div>
                        <div class="form-group">
                            {{ form.email.label(class="form-control-label") }}
                            {% if form.email.errors %}
                                {{ form.email(class="form-control form-
control-lg is-invalid") }}
                                <div class="invalid-feedback">
                                    {% for error in form.email.errors %}
                                        <span>{{ error }}</span>
                                    {% endfor %}
                                </div>
                            {% else %}
                                {{ form.email(class="form-control form-
control-lg") }}
                            {% endif %}
                        </div>
                        <div class="form-group">
                            {{ form.password.label(class="form-control-label")
}}
                            {% if form.password.errors %}
                                {{ form.password(class="form-control form-
control-lg is-invalid") }}
```

```
                                <div class="invalid-feedback">
                                    {% for error in form.password.errors %}
                                        <span>{{ error }}</span>
                                    {% endfor %}
                                </div>
                            {% else %}
                                {{ form.password(class="form-control form-
control-lg") }}
                            {% endif %}
                        </div>
                        <div class="form-group">
                            {{ form.confirm_password.label(class="form-
control-label") }}
                            {% if form.confirm_password.errors %}
                                {{ form.confirm_password(class="form-control
form-control-lg is-invalid") }}
                                <div class="invalid-feedback">
                                    {% for error in
form.confirm_password.errors %}
                                        <span>{{ error }}</span>
                                    {% endfor %}
                                </div>
                            {% else %}
                                {{ form.confirm_password(class="form-control
form-control-lg") }}
                            {% endif %}
                        </div>
                    </fieldset><br>
                    <div class="form-group">
                        {{ form.submit(class="btn btn-primary") }}
                    </div> <br>
                    <a href="{{ url_for('glogin') }}"><button
type="button"  class="btn btn-primary btn-floating mx-1">
                      Sign up using Google
                    </button></a>
                </form>
                </div>
              </div>
            </div>
        </div>
      </div>
    </section>
  </body>
</html>
```

```
Prediction.html
<!DOCTYPE html>
<!--=== Coding by CodingLab | www.codinglabweb.com === -->
```

```
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <!----======== CSS ======== -->
    <link rel="stylesheet" href="{{ url_for('static', filename='main.css')
}}">

    <!----===== Iconscout CSS ===== -->
    <link rel="stylesheet"
href="https://unicons.iconscout.com/release/v4.0.0/css/line.css">

    <title>Prediction Form </title>
</head>
<body>
    <div class="container">
        <header>Enter your details.</header>

        <form action="{{ url_for('result') }}" method="POST">
            <div class="form first">
                <div class="details personal">
                    <span class="title">kidney Details</span>

                    <div class="fields">
                        <div class="input-field">
                            <label>Age</label>
                            <input type="number" name="age" placeholder="34
(age in years)" required>
                        </div>

                        <div class="input-field">
                            <label>Enter Urine Specific gravity</label>
                            <input type="text" name="usg" placeholder="1.005
(SG)" required>
                        </div>

                        <div class="input-field">
                            <label>Enter serum albumin level</label>
                            <input type="text" name="sal" placeholder="3.4
(g/dL)" required>
                        </div>

                        <div class="input-field">
                            <label>Enter Blood pressure</label>
                            <input type="text" name="bp" placeholder="90
(mm/hg)" required>
```

```
                        </div>

                        <div class="input-field">
                            <label>Enter Sugar</label>
                            <input type="text" name="su" placeholder="4.0
(mmol/L)" required>
                        </div>

                        <div class="input-field">
                            <label>Enter Blood urea level</label>
                            <input type="text" name="bu" placeholder="65
(mg/dl)" required>
                        </div>

                        <div class="input-field">
                            <label>Enter Serum-creatinine level</label>
                            <input type="text" name="sc" placeholder="1.2
(mg/dl)" required>
                        </div>

                        <div class="input-field">
                            <label>Enter Sodium level</label>
                            <input type="text" name="sod" placeholder="135
(mEq/L)" required>
                        </div>

                        <div class="input-field">
                            <label>Enter potassium level</label>
                            <input type="text" name="pot" placeholder="5.7
(mEq/L)" required>
                        </div>

                        <div class="input-field">
                            <label>Enter Hemoglobin</label>
                            <input type="text" name="hg" placeholder="13.8
(g/dL)" required>
                        </div>

                        <div class="input-field">
                            <label>Enter Packed cell volume</label>
                            <input type="text" name="pcv" placeholder="44 (%)"
required>
                        </div>

                        <div class="input-field">
                            <label>Enter white blood cell count</label>
                            <input type="text" name="wbcc" placeholder="4000
WBCs per microliter" required>
```

```
                    </div>

                    <div class="input-field">
                        <label>Diabetesmellitus</label>
                        <select  name="diab" required>
                            <option disabled selected>Select</option>
                            <option>yes</option>
                            <option>no</option>
                        </select>
                    </div>

                    <div class="input-field">
                        <label>coronary artery disease</label>
                        <select  name="cad" required>
                            <option disabled selected>Select</option>
                            <option>yes</option>
                            <option>no</option>
                        </select>
                    </div>

                    <div class="input-field">
                        <label>HyperTension</label>
                        <select  name="hp" required>
                            <option disabled selected>Select</option>
                            <option>yes</option>
                            <option>no</option>
                        </select>
                    </div>
                </div>
            </div>
            <div>
                <button class="sumbit">
                    <span class="btnText">Submit</span>
                    <i class="uil uil-navigator"></i>
                </button>
                </div>
            </div>
        </div>
    </form>
</div>

<script src="{{ url_for('static', filename='script.js') }}"></script>
</body>
</html>
Forgotpassword.html
<!DOCTYPE html>
<html lang="en">
```

```
<head>

    <title>CODE WITH HOSSEIN</title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <!-- Bootstrap v5.1.3 CDNs -->
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css"
rel="stylesheet">
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min
.js"></script>
    <link rel="stylesheet"
href="https://use.fontawesome.com/releases/v5.0.8/css/all.css">

    <!-- CSS File -->
    <style>
      * {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}

body {
    height: 100vh;
    display: flex;
    align-items: center;
    justify-content: center;
    background: #F0FFFF;
}

.login {
    width: 500px;
    height: min-content;
    padding: 20px;
    border-radius: 12px;
    background: #ffffff;
}

.login h1 {
    font-size: 36px;
    margin-bottom: 25px;
}

.login form {
    font-size: 20px;
}
```

```
.login form .form-group {
    margin-bottom: 12px;
}

.login form input[type="submit"] {
    font-size: 20px;
    margin-top: 15px;
}

    .btn-primary {
    color: #fff;
    background-color: #2487ce;
    border-color: #2487ce;
}
    </style>

</head>

<body>

    <div class="login" style="font-family:Georgia;">

        <h1 class="text-center" style="font-family:Georgia;font-size:
25px">Trouble Logging in?</h1>
        <div class="text-center" style="font-family:Georgia;font-size:
17px">Enter your email id we will send you a link to get back into your
account.</div>

        <form method="POST" action="">
            {{ form.hidden_tag() }}
            <fieldset class="form-group">
                <legend class="border-bottom mb-4">Reset Password</legend>
                <div class="form-group">
                    {{ form.email.label(class="form-control-label") }}
                    {% if form.email.errors %}
                        {{ form.email(class="form-control form-control-lg is-
invalid") }}
                        <div class="invalid-feedback">
                            {% for error in form.email.errors %}
                                <span>{{ error }}</span>
                            {% endfor %}
                        </div>
                    {% else %}
                        {{ form.email(class="form-control form-control-lg") }}
                    {% endif %}
                </div>
            </fieldset>
            <div class="form-group">
```

```
                {{ form.submit(class="btn btn-primary") }}
            </div>
        </form>

    </div>

</body>

</html>


Changepassword.html
<!DOCTYPE html>
<html lang="en">

<head>

    <title>Reset Password</title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <!-- Bootstrap v5.1.3 CDNs -->
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css"
rel="stylesheet">
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min
.js"></script>
    <link rel="stylesheet"
href="https://use.fontawesome.com/releases/v5.0.8/css/all.css">

    <!-- CSS File -->
    <style>
      * {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}

body {
    height: 100vh;
    display: flex;
    align-items: center;
    justify-content: center;
    background: #F0FFFF;
}

.login {
    width: 500px;
    height: min-content;
```

```
    padding: 20px;
    border-radius: 12px;
    background: #ffffff;
}

.login h1 {
    font-size: 36px;
    margin-bottom: 25px;
}

.login form {
    font-size: 20px;
}

.login form .form-group {
    margin-bottom: 12px;
}

.login form input[type="submit"] {
    font-size: 20px;
    margin-top: 15px;
}

    .btn-primary {
    color: #fff;
    background-color: #2487ce;
    border-color: #2487ce;
}
    </style>

</head>

<body>

    <div class="login" style="font-family:Georgia;">

        <h1 class="text-center" style="font-family:Georgia;font-size:
25px">Change password</h1>

        <form method="POST" action="">
            {{ form.hidden_tag() }}
            <fieldset class="form-group">
                <legend class="border-bottom mb-4">Reset Password</legend>
                <div class="form-group">
                    {{ form.password.label(class="form-control-label") }}
                    {% if form.password.errors %}
                        {{ form.password(class="form-control form-control-lg
is-invalid") }}
```

```
                        <div class="invalid-feedback">
                            {% for error in form.password.errors %}
                                <span>{{ error }}</span>
                            {% endfor %}
                        </div>
                    {% else %}
                        {{ form.password(class="form-control form-control-lg")
}}
                    {% endif %}
                </div>
                <div class="form-group">
                    {{ form.confirm_password.label(class="form-control-label")
}}
                    {% if form.confirm_password.errors %}
                        {{ form.confirm_password(class="form-control form-
control-lg is-invalid") }}
                        <div class="invalid-feedback">
                            {% for error in form.confirm_password.errors %}
                                <span>{{ error }}</span>
                            {% endfor %}
                        </div>
                    {% else %}
                        {{ form.confirm_password(class="form-control form-
control-lg") }}
                    {% endif %}
                </div>
            </fieldset>
            <div class="form-group">
                {{ form.submit(class="btn btn-outline-info") }}
            </div>
        </form>

    </div>

</body>

</html>
```

```
Blog.html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Useful links</title>
```

```
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
1BmE4kWBq78iYhFldvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoqyl2QvZ6jIW3"
crossorigin="anonymous"/>
    <link rel="stylesheet"
href="https://use.fontawesome.com/releases/v5.0.8/css/all.css">
    <style>
    .form-control form-control-lg{
      font-size: 18px;
    }
    .btn-primary {
    color: #fff;
    background-color: #2487ce;
    border-color: #2487ce;
}
</style>

  </head>
  <body style="font-family: 'Times New Roman', Times, serif;" >
    <section>
      <div class="jumbotron text-center"><h1><strong><b>Understanding
CKD</b></strong></h1> </div>
      <div class = "container" style="margin-top:50px;">
        <div class="row">

      <div class="card border-info mb-3" style="max-width: 18rem;margin-right:
50px;">
  <div class="card-header" style="max-width: 18rem;">CKD BASICS</div>
  <div class="card-body">
    <h5 class="card-title">What is CKD?</h5>
    <p class="card-text">Chronic kidney disease, also known as chronic renal
disease or CKD, is a condition....</p>
    <a href="https://www.kidney.org/atoz/content/about-chronic-kidney-disease"
class="btn btn-primary">Learn more</a>
  </div>
</div>



      <div class="card border-info mb-3" style="max-width: 18rem;margin-right:
50px;">
  <div class="card-header" style="max-width: 18rem;">Kidney health
testing</div>
  <div class="card-body">
    <h5 class="card-title">glomerular filtration rate (GFR)</h5>
    <p class="card-text">The glomerular filtration rate (GFR) shows how well
the kidneys are filtering...</p>
```

```
    <a href="https://www.kidney.org/atoz/content/gfr" class="btn btn-
primary">Learn more</a>
  </div>
</div>


      <div class="card border-info mb-3" style="max-width: 18rem;margin-right:
50px;">
  <div class="card-header" style="max-width: 18rem;">Kidney health
testing</div>
  <div class="card-body">
    <h5 class="card-title">Understanding Your Lab Values</h5>
    <p class="card-text">People who develop chronic kidney disease may have
some or all of the following tests and measurements...</p>
    <a href="https://www.kidney.org/atoz/content/understanding-your-lab-
values" class="btn btn-primary">Learn more</a>
  </div>
</div>

<div class="card border-info mb-3" style="max-width: 18rem;">
  <div class="card-header" style="max-width: 18rem;">Stages of CKD</div>
  <div class="card-body">
    <h5 class="card-title">What are the Stages of Chronic Kidney Disease?</h5>
    <p class="card-text">If you have Chronic Kidney Disease (CKD) or know
someone who does you may want to note the various stages...</p>
    <a href="https://www.kidney.org/blog/kidney-cars/what-are-stages-chronic-
kidney-disease" class="btn btn-primary">Learn more</a>
  </div>
</div>

       </div>

    <div class="row">

      <div class="card border-info mb-3" style="max-width: 18rem;margin-right:
50px;">
  <div class="card-header" style="max-width: 18rem;">Risk factors &
causes</div>
  <div class="card-body">
    <h5 class="card-title">Causes of CKD</h5>
    <p class="card-text">Chronic kidney disease includes conditions that
damage your kidneys and decrease their ability...</p>
    <a href="https://www.kidney.org/atoz/content/about-chronic-kidney-disease"
class="btn btn-primary">Learn more</a>
  </div>
</div>
```

```
      <div class="card border-info mb-3" style="max-width: 18rem;margin-right:
50px;">
  <div class="card-header" style="max-width: 18rem;">Risk factors &
causes</div>
  <div class="card-body">
    <h5 class="card-title">High blood pressure (hypertension)</h5>
    <p class="card-text">High blood pressure (hypertension) is a leading cause
of kidney disease...</p>
    <a href="https://www.kidney.org/atoz/atozTopic_HighBloodPressure"
class="btn btn-primary">Learn more</a>
  </div>
</div>


      <div class="card border-info mb-3" style="max-width: 18rem;margin-right:
50px;">
  <div class="card-header" style="max-width: 18rem;">Risk factors &
causes</div>
  <div class="card-body">
    <h5 class="card-title">Heart Disease</h5>
    <p class="card-text">Your heart and kidneys are two important organs in
your body. They work together...</p>
    <a href="https://www.kidney.org/atoz/content/heart-and-kidney-connection"
class="btn btn-primary">Learn more</a>
  </div>
</div>

<div class="card border-info mb-3" style="max-width: 18rem;">
  <div class="card-header" style="max-width: 18rem;">Risk factors &
causes</div>
  <div class="card-body">
    <h5 class="card-title">What is Diabetes?</h5>
    <p class="card-text">Diabetes is a condition in which your body has
trouble controlling...</p>
    <a href="https://www.kidney.org/atoz/atozTopic_Diabetes" class="btn btn-
primary">Learn more</a>
  </div>
</div>

        </div>

    <div class="row">

      <div class="card border-info mb-3" style="max-width: 18rem;margin-right:
50px;">
  <div class="card-header" style="max-width: 18rem;">Nutrition</div>
  <div class="card-body">
```

```
    <h5 class="card-title">Nutrition and Early Kidney Disease (Stages 1-
4)</h5>
    <p class="card-text">Making healthy food choices is important to us all,
but it is even more essential if you have kidney disease (CKD)...</p>
    <a href="https://www.kidney.org/atoz/content/nutrikidfail_stage1-4"
class="btn btn-primary">Learn more</a>
  </div>
</div>



      <div class="card border-info mb-3" style="max-width: 18rem;margin-right:
50px;">
  <div class="card-header" style="max-width: 18rem;">Prevention</div>
  <div class="card-body">
    <h5 class="card-title">7 Golden Rules of Kidney Disease Prevention</h5>
    <p class="card-text">Many of us don't give much thought to our hardworking
kidneys. The truth is 33% of adults in the United States are at risk for
developing kidney disease...</p>
    <a href="https://www.kidney.org/prevention/7-golden-rules-of-prevention"
class="btn btn-primary">Learn more</a>
  </div>
</div>



      <div class="card border-info mb-3" style="max-width: 18rem;margin-right:
50px;">
  <div class="card-header" style="max-width: 18rem;">Prevention</div>
  <div class="card-body">
    <h5 class="card-title">foods to Avoid</h5>
    <p class="card-text">Your kidneys are organs that play several important
roles in your health...</p>
    <a href="https://www.healthline.com/nutrition/foods-to-avoid-with-kidney-
disease-and-diabetes" class="btn btn-primary">Learn more</a>
  </div>
</div>

<div class="card border-info mb-3" style="max-width: 18rem;">
  <div class="card-header" style="max-width: 18rem;">Prevention</div>
  <div class="card-body">
    <h5 class="card-title">Nutrition, Diet & Exercise</h5>
    <p class="card-text"> healthy diet and exercise are important parts of
living well with kidney disease. Your nutrition needs...</p>
    <a href="https://www.kidney.org/category/diet-nutrition-exercise"
class="btn btn-primary">Learn more</a>
  </div>
</div>
```

```
        </div>



  </body>
  </html>
```

```
Result.html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css"
rel="stylesheet">
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min
.js"></script>
    <link rel="stylesheet"
href="https://use.fontawesome.com/releases/v5.0.8/css/all.css">
    <title>CKD Predicted Category</title>
    <style>body {
        height: 100vh;
        display: flex;
        align-items: center;
        justify-content: center;
        background: #F0FFFF;
    }</style>
</head>
<body>
    <div class="container" style="width: 50%; height: 400px;">



          <h1 class="card-text" style="text-align:center;font-size: xx-
large;font-family: Verdana, Geneva, Tahoma, sans-serif;"><strong>You are at a
risk for CKD</strong></h1>

    </div>

</body>
</html>
```

GITHUB & PROJECT DEMO LINK:

Github Link:

https://github.com/IBM-EPBL/IBM-Project-22544-1659853727

Demo Link:

https://youtu.be/fDtmYIM9zt0