Project Development Phase

Sprint 4

Date	17 November 2022
Team ID	PNT2022TMID53061
Project Name	Project - Personal Expense Tracker
Team Members	Jagadish R, Karthikraja A, Jeet Golecha M, Murali Krishnan R

Code

Updated on **GitHub** in the **Sprint-4** folder under Project Development Phase.

Description of USN and Screenshots
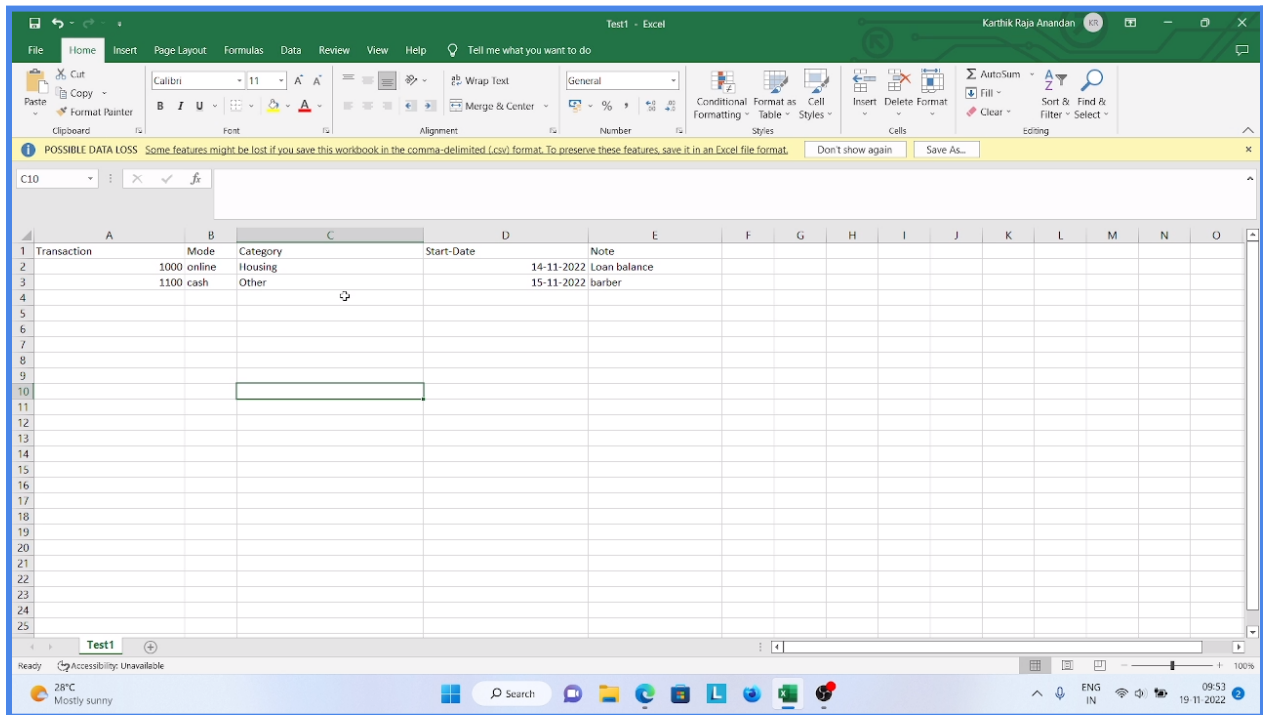
USN-10

As a user, I should be provided with the flexibility to provide inputs in multiple file formats.
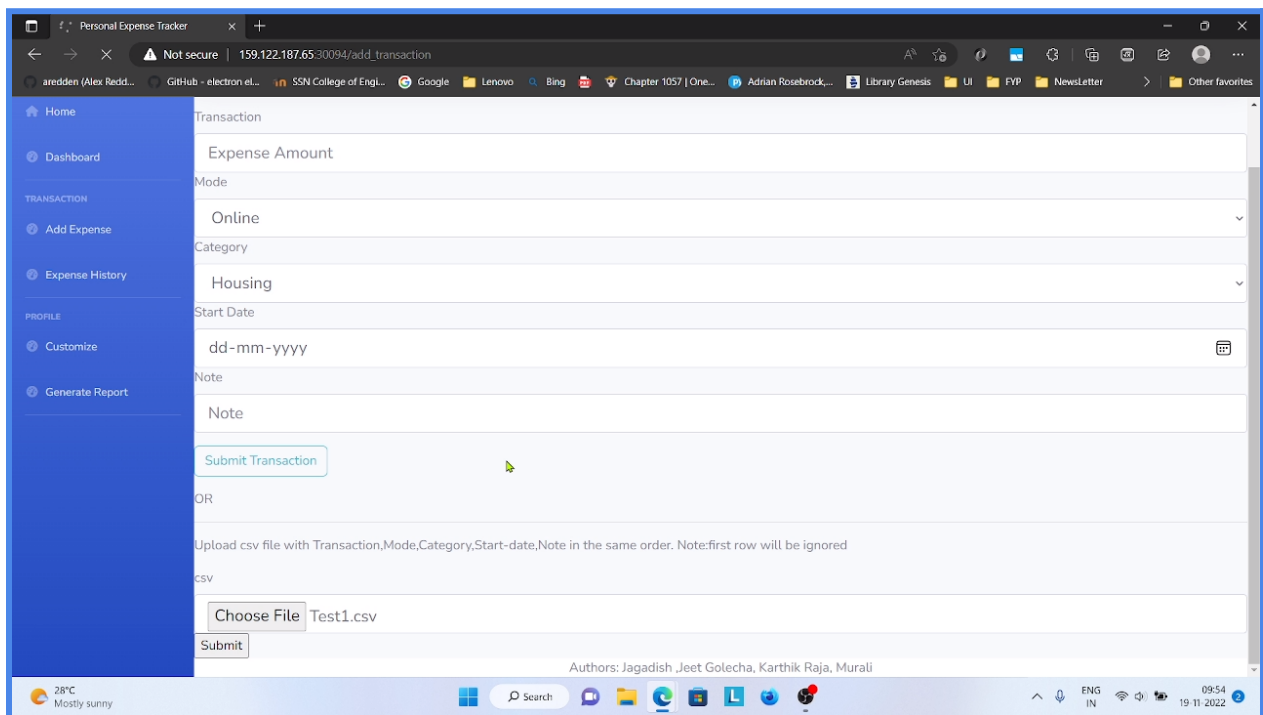
Multiple Input File Formats

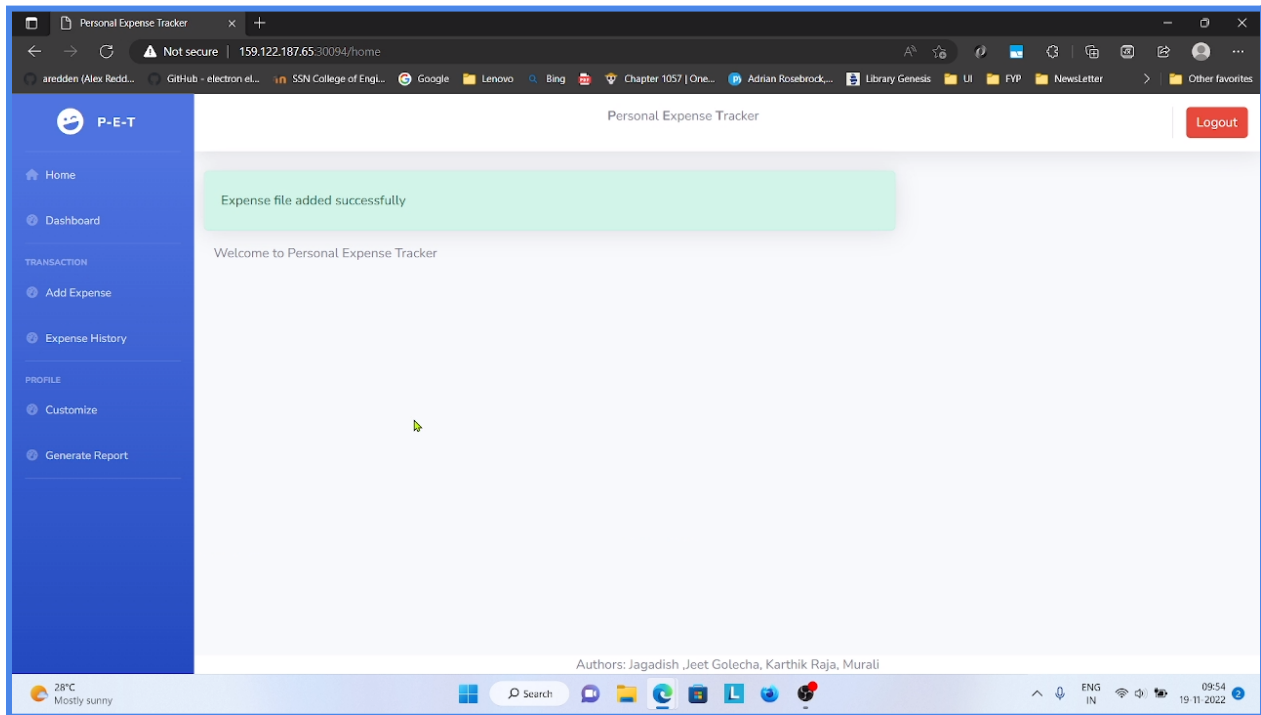● User can have option of sending csv file containing transactions in them

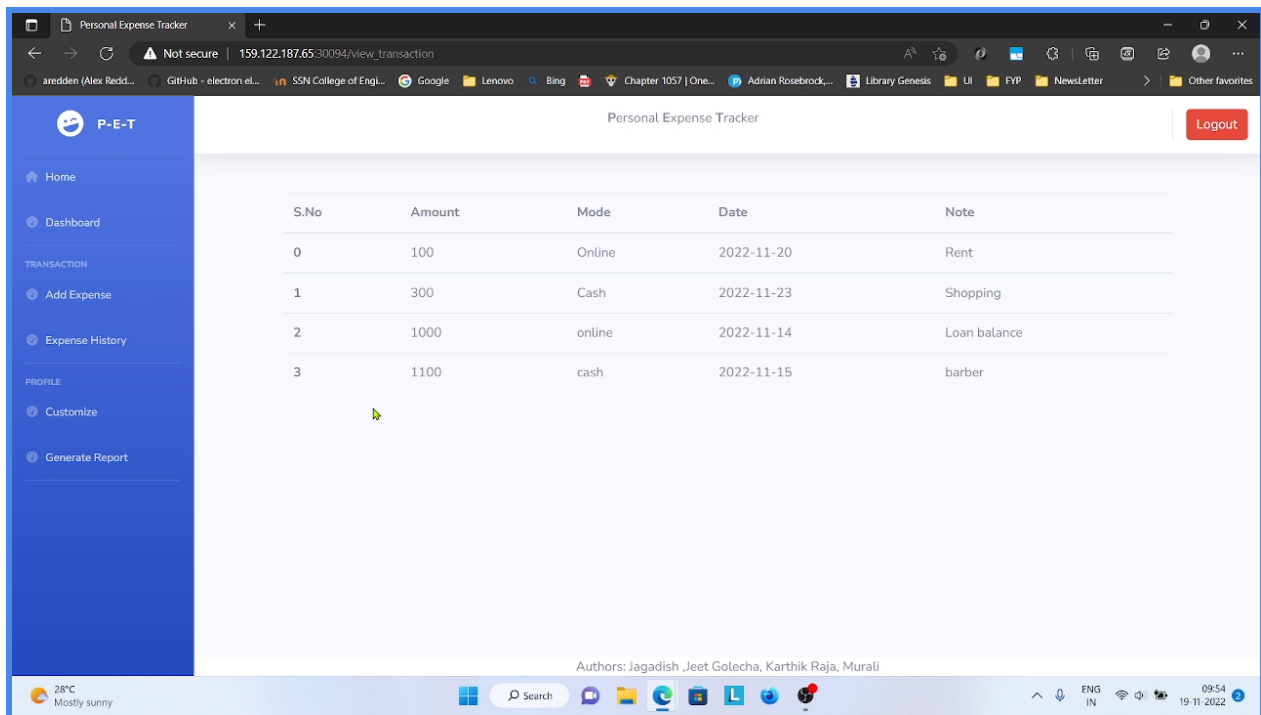- Assume the user have their transactions in csv

- The csv file added to page

- The transactions are added to db2 successfully

- When user checks the expense history, their transactions in the csv are displayed in the page

USN-11

As a user, I should be able to access the application from anywhere around the world.

Deployment to IBM Cloud

- The app is deployed in the IBM cloud so that user can access the application from anywhere around the world

Docker

- The docker image is built using Dockerfile
- The all necessary files are included in the image

```
# Build the Docker image
- name: Build with Docker
  run: |
    docker build -t "$REGISTRY_HOSTNAME"/"$ICR_NAMESPACE"/"$IMAGE_NAME":latest .
    ibmcloud cr login

# Push the image to IBM Container Registry
- name: Push the image to ICR
  run: |
    ibmcloud cr login
    ibmcloud cr image-rm "$ICR_NAMESPACE"/"$IMAGE_NAME"
    docker push $REGISTRY_HOSTNAME/$ICR_NAMESPACE/$IMAGE_NAME:latest
```

```
✓ Build with Docker                                                                    1m 20s
202  WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the
     system package manager. It is recommended to use a virtual environment instead:
     https://pip.pypa.io/warnings/venv
203
204  Notice:  A new release of pip available: 22.3 -> 22.3.1
205  Notice:  To update, run: pip install --upgrade pip
206  Removing intermediate container dfa388d7dcac
207   ---> 030fbbdf01b1
208  Step 5/8 : COPY . /flaskApp
209   ---> b4937dbfae13
210  Step 6/8 : ENTRYPOINT [ "python" ]
211   ---> Running in 7e9909c79fb4
212  Removing intermediate container 7e9909c79fb4
213   ---> f0d6f9dd6e44
214  Step 7/8 : EXPOSE 5000
215   ---> Running in 31304e1a7a10
216  Removing intermediate container 31304e1a7a10
217   ---> e33b9afba96e
218  Step 8/8 : CMD ["app.py" ]
219   ---> Running in b8e0c9e0cca3
220  Removing intermediate container b8e0c9e0cca3
221   ---> fabb16afa7de
222  Successfully built fabb16afa7de
223  Successfully tagged icr.io/***/iks-test:latest
224  Logging 'docker' in to 'icr.io'...
225  Logged in to 'icr.io'.
226
```

IBM Container Registry

- The created docker image is pushed to ibm container registry
- The registry now contains our image file

```
✓  Push the image to ICR                                                               32s

 1  ▶ Run ibmcloud cr login
21  Logging 'docker' in to 'icr.io'...
22  Logged in to 'icr.io'.
23
24  OK
25  Deleting image '***/iks-test'...
26
27  Successfully deleted image 'sha256:0893b46151f667cce540f3cca98fc3bfca939c92a558cfff28532fddb28998ab'
28
29  OK
30  The push refers to repository [icr.io/***/iks-test]
31  a239be25e4e9: Preparing
32  ed6bedeb2361: Preparing
33  b9f246e86b79: Preparing
34  345c9e42b8e4: Preparing
35  24bf8dd8c4a6: Preparing
36  18bbb218c890: Preparing
37  e6e9854ca999: Preparing
```

IBM Kubernetes

- The pods are created using deployment with our image.
- Then, the service is created for the deployments to connect with external systems.
- The service is then exposed to nodeport.
- The public IP address and the nodeport gives the public url where our app is deployed.

```
# Deploy the Docker image to the IKS cluster
- name: Deploy to IKS
  run: |
    ibmcloud ks cluster config --cluster $IKS_CLUSTER
    kubectl config current-context
    kubectl delete deploy $DEPLOYMENT_NAME
    kubectl create deployment $DEPLOYMENT_NAME --image=$REGISTRY_HOSTNAME/$ICR_NAMESPACE/$IMAGE_NAME:latest
    kubectl delete svc $DEPLOYMENT_NAME
    kubectl expose deployment/$IMAGE_NAME --type="NodePort" --port 5000
    kubectl set env deployment/$DEPLOYMENT_NAME  DB_API_UID=${{ secrets.DB_API_UID }}
    kubectl set env deployment/$DEPLOYMENT_NAME  DB_API_PWD=${{ secrets.DB_API_PWD }}
    kubectl set env deployment/$DEPLOYMENT_NAME DB_API_HOST=${{ secrets.DB_API_HOST }}
    kubectl set env deployment/$DEPLOYMENT_NAME DB_API_PORT=${{ secrets.DB_API_PORT }}
    kubectl set env deployment/$DEPLOYMENT_NAME  SENDGRID_API_KEY=${{ secrets.SENDGRID_API_KEY }}
    kubectl describe services/$IMAGE_NAME
    ibmcloud cs workers --cluster $IKS_CLUSTER
```

```
  ✓  Deploy to IKS                                                                           13s
  49  Labels:                  app=iks-test
  50  Annotations:             <none>
  51  Selector:                app=iks-test
  52  Type:                    NodePort
  53  IP Family Policy:        SingleStack
  54  IP Families:             IPv4
  55  IP:                      172.21.56.106
  56  IPs:                     172.21.56.106
  57  Port:                    <unset>  5000/TCP
  58  TargetPort:              5000/TCP
  59  NodePort:                <unset>  30094/TCP
  60  Endpoints:               <none>
  61  Session Affinity:        None
  62  External Traffic Policy: Cluster
  63  Events:                  <none>
  64  OK
  65  ID                                                    Public IP       Private IP     Flavor   State     Status
      Zone    Version
  66  kube-cdm8psef0s018qv50fv0-petfree-default-0000004c    159.122.187.65  10.144.185.9   free     normal    Ready
      mil01   1.24.7_1543
```

USN-12

As a user, I can launch the application and add or manage the expenses and get notified for over spent.

App Url -> http://159.122.187.65:30094/

Launch Application

- The user can now launch the application.
- The user can add or manage expense and be notified for over spent.