PERSONAL EXPENSE TRACKER APPLICATION

IBM-Project-27235-1660051509- (PNT2022TMID06598)

NALAIYATHIRAN PROJECT BASED LEARNING ON PROFESSIONAL READINESS FOR INNOVATION,EMPLOYMENT AND ENTREPRENEURSHIP.

PROJECTREPORT

- **Sowndariyalakshmi C(730419205048)**
- **Gnana RethesS(730419205010)**
- **Gogularam S P(730419205011)**
- **Nandhini V(730419205028)**

BACHELOR OF TECHNOLOGY INFORMATIONTECHNOLOGY

INDEX

3. DatabaseSchema(ifApplicable)

8. **TESTING**

1. TestCases
2. UserAcceptanceTesting

9. **RESULTS**

1. Performance Metrics

10. **ADVANTAGES&DISADVANTAGES**

11. **CONCLUSION**

12. **FUTURE SCOPE**

13. **APPENDIX**

SourceCode

GitHub &ProjectDemoLink

1. Introduction:

1.1. Project

overview:Category: Cloud App

TeamID:PNT2022TMID06598

💻Skills Required: IBM Cloud, HTML, Javascript, IBM Cloud Object Storage, Python Flask, Kubernetes,Docker,IBM DB2,IBM Container Registry

In simple words, personal finance entails all the financial decisions and activitiesthat a Finance app makes your life easier by helping you to manage your finances efficiently. Apersonal finance app will not only help you with budgeting and accounting but also give youhelpful insights about money management. Personal finance applications will ask users to addtheir expenses and based on their expenses wallet balance will be updated which will be visibletotheuser.Also,userscangetananalysisoftheirexpenditureingraphicalforms.Theyhaveanoption to set a limit for the amount to be used for that particular month if the limit is exceededthe user will be notified with an email alert.

1.2. Purpose:

It's easy to make this part of your everyday routine thanks to expense tracker appsthat help you manage your money on the go. These apps certainly overlap with budgeting apps,but while the latter provides a big-picture view of your finances, expense tracker apps put moreof an emphasis on your spending. These apps usually categorize your expenses and help you getagood idea of your purchasing behavior.

Thedatabaseconnectivityisplannedusingthe"SQLConnection"methodology.Thestandardsof security and data protective mechanisms have been given a big choice for proper usage.The application takes careof different module sand their associated reports" which are produced as

Per the applicable strategies and standards that are put forward by the administrative staff.

The entire project has been developed keeping in view of the distributed client server computing technology" in mind. The specification has been normalized up to 3NF to eliminate all the
anomalies that may arise due to the database transactions that are executed by the general users and the organizational administration.The user interfaces are browser specific to give distributed accessibility for the overall system. The internal database has been selected as MS-SQL server2000.

The basic constructs of table spaces" clusters and inde0es have been exploited to provide higher consistency and reliability for the data storage. TheMS-SQL server 2000 was a choice asit provides the constructs of high-level reliability and security. The total front end was dominated using the A%(.)et technologies. At all proper levels high care was taken to check thatthe system manages the data consistency with proper business rules or validations.

The database connectivity was planned using the latest "SQL Connection" technology provided by Microsoft corporation. The authentication and authorization was cross checked atall there levant stages.The user level accessibility has been restricted into two zones namely.

2. LiteratureSurvey:

2.1. Existingproblem:

Students write their expenses in a notebook and find it difficult infinding the major expenses category so it tends them to expend more. Studentsmay miss out on writing their expenses in the notebook, which makes themfrustrated. Family guardians take the monthly grocery list on a piece of paper sothey lose that they get frustrated. Computer users track their expenses in an excelsheetbyusingitandgetboredwiththerowsandcolumns.MobileUsersmanagethei r expenses in a mobile app they spend more if it doesn't have daily limit remainderssothey get disappointment easily.

2.2. References:

[1] Velmurugan;RichardFrancis;*Expensemanagerapplication.Dec2020*

[2] Saumya Dubey; Rigved Rishabh Kumar; *Student expense trackingapplication. Apr 2014*

[3] JosephJofish;Rebecca;*TrackingpersonalfinancesApr2014*

[4] Shahed Anzarus Sabab; Sadman Saumik Islam, Md. Jewel Rana; MonirHossian;*e-Expense:ASmartApproachtoTrackEverydayExpense*

[5] YashveerSingh;DhirendraYadav;KunalSingh;*GroupexpensetrackerJun2020*

[6] DailyExpenseTrackerMobileApplication;NuuraNajatiBintiMustafa

[7] Expendituremanagementsystem;Dr.V.Geetha,G.Nikhitha.[8]

. Daily Expense Tracker; Shivam Mehra, Prabhat

Parashar[9].Expensetrackerapplication;Velmurugan.R,Mrs.P.U

sha

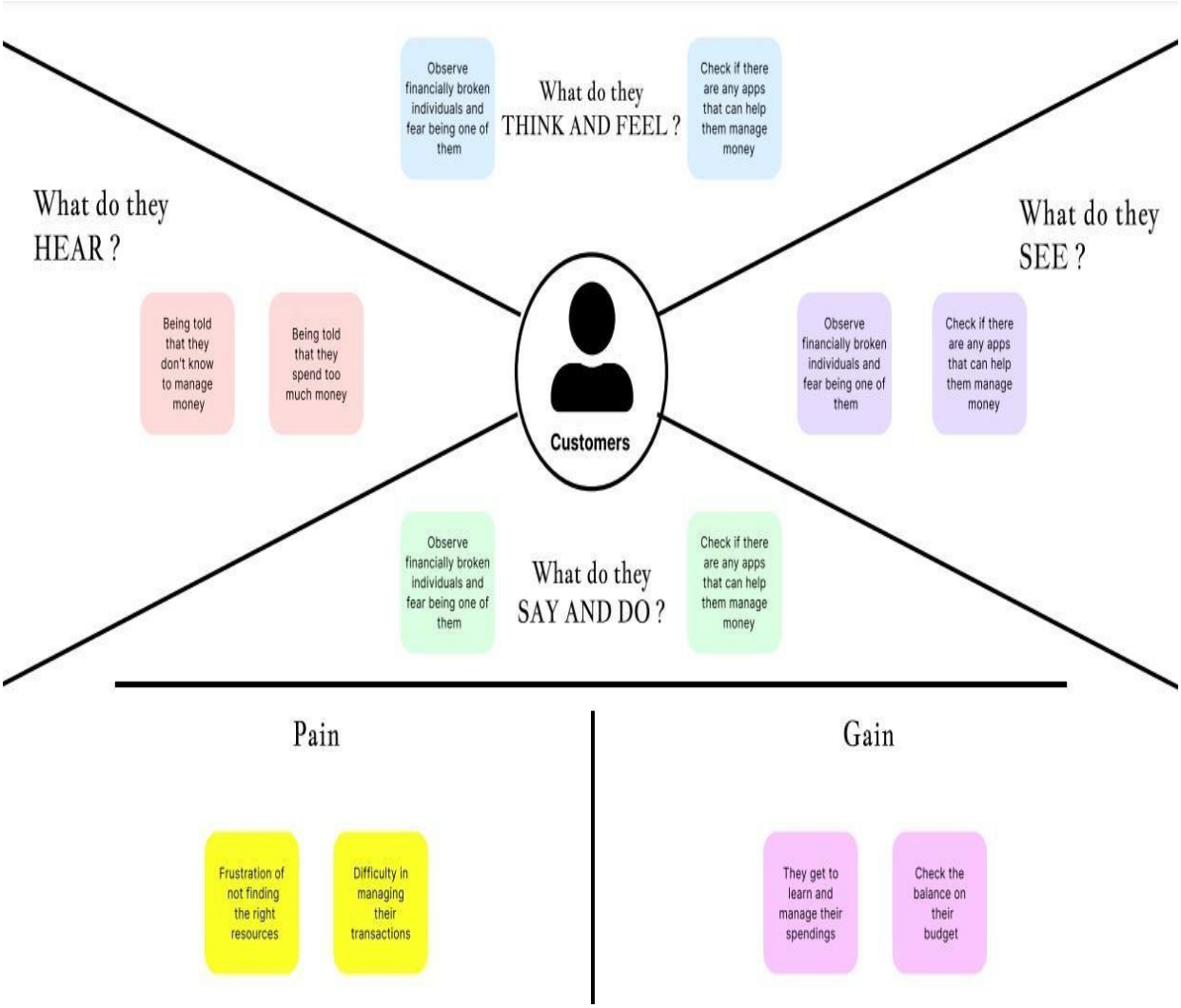
[10].IntelligentOnlineBudgetTracker;GirishBekarooand SameerSunhaloo

2.3. Problemstatementdefinition:

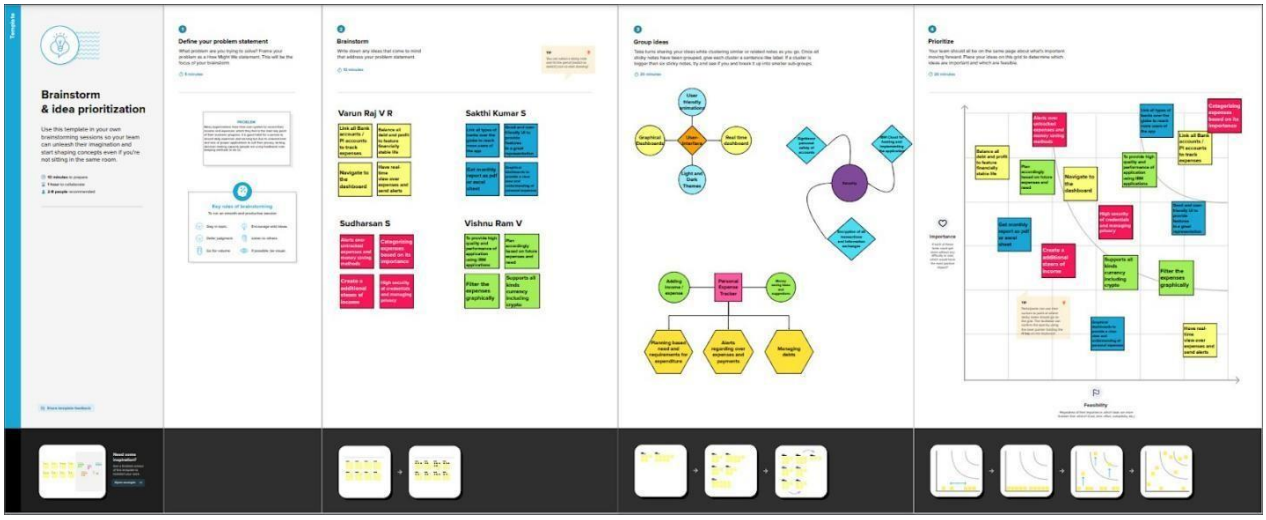
Manyorganizationshavetheirownsystemtorecordtheirincomeandexpenses,which they feel is the main key point of their business progress. It is good habit for aperson to record daily expenses and earning but due to unawareness and lack of proper applicationsto suit their privacy, lacking decision making capacity people are using traditional notekeeping methods to do so.

3. IDEATION&PROPOSEDSOLUTION

3.1. Empathymapcanvas:



3.2. Ideation&Brainstorming:



3.3. **ProposedSolution:**

- Our application requests users to add their expenses and based on their expenses walletbalancewillbeupdatedwhichwillbevisibletotheuser.Userscanavailanoptiontoseta limit for the amount to be used for that particular cycle and when the user exceeds thelimit, he receives an alert.
- The application also as helps to keep track of bills the user has to pay, so the user getsregularremindersofduedatesofbillpayment.Wealsomightsendtheuseroccasionalnotifications on how much limit he has left and his due dates.
- The user will be able to Stick to their Spending Limits. They can able to scan their billsany time thus data loss is avoided.
- Users can keep track of credit card bills and make payments on time so as to not get any
- unwanted interests.

3.4. ProblemSolutionFit:

Project Title: Personal Expense Tracker

Project Design Phase-I - Solution Fit Template

Team ID: PNT2022TMID24728

Define CS, fit into CC

1. CUSTOMER SEGMENT(S) CS

Our customers are people who have trouble tracking their expenses and would like to have an interactive application that shows them constant reminders about spending limits and due dates of bill payments.

6. CUSTOMER CONSTRAINTS CC

Customers may find trouble regarding adding their income and expenses, as they have to enter every record manually. Sometimes the reminders and alerts about their budgets and goals might become overwhelming and annoying. Users also must have certain financial knowledge for using the application.

5. AVAILABLE SOLUTIONS AS

While tracking expenses manually, customers have to track it and do calculations manually. So our application aims at removing that problems.

Pros:

1. Users can add their income and expenses and the app calculates the remaining balance, total amount spent by category and much more.
2. Users can add their payment dues and also their last dates so they will get reminders and alerts.

Cons:

1. Manually enter every single transaction.

Explore AS, differentiate

2. JOBS-TO-BE-DONE / PROBLEMS J&P

In paper-based expense tracker system it is difficult to track our monthly expenses manually. The paper-based expense records may get lost very easily or even destroyed. Also people tend to forget their payment dates sometimes and get charged with late payment fees and interests.

9. PROBLEM ROOT CAUSE RC

Written records are volatile and can be sometimes illegible as handwriting differs according to each person. So it can be hard to understand. Bill payment should be remained constantly as people forget about it during their daily life hustles.

7. BEHAVIOUR BE

They may keep a temporary note on their mobile. People should know their budgets for each month and spend appropriately according to their goals.

Manually track bills and payment dates maybe using sticky notes.

Identify strong TR & EM

3. TRIGGERS TR

- Realizing that excessive spending leading to lack of money in case of emergencies
- Lack of Budgeting knowledge.

10. YOUR SOLUTION SL

- A cloud-based web application which keeps track of user's personal expenses This system attempts to free the user with much of the burden of manual calculation and to keep track of the expenditure.
- Users just need to enter their day-to-day expenses. They also have an option to set their limit. If the expenditure exceeds that limit, notification will be sent through mail.

8. CHANNELS of BEHAVIOR CH

8.1 ONLINE

Less security and customer support. Real-time notification for un-tracked expenses is not available

8.2 OFFLINE

Email alerts are disabled in offline mode as users are not connected to the internet.

Identify strong TR & EM

4. EMOTIONS: BEFORE / AFTER EM

Before: Frustration, Confusion, Inadequate

After: Boost , Feeling smart , Be an example for others

- Application enables users to add and track their bills and receive regular alerts on their due dates

4. Requirementanalysis:

4.1Functionalrequirement:

FRNo.	Functional Requirement(Epic)	SubRequirement(Story/Sub-Task)
FR-1	UserRegistration	RegistrationthroughFormandGoogleOAuth
FR-2	UserConfirmation	EmailOTPverification
FR-3	UserFinancialAccounts	Userdataentry
FR-4	UserDashboard	ExpenseData,setgoals,addincomesandaddbills
FR-5	UserNotifications	SystemAccess

		RealtimeAlerting
FR-6	SecurityofUserData	SecuredDatabase DataSecurityAlgorithms

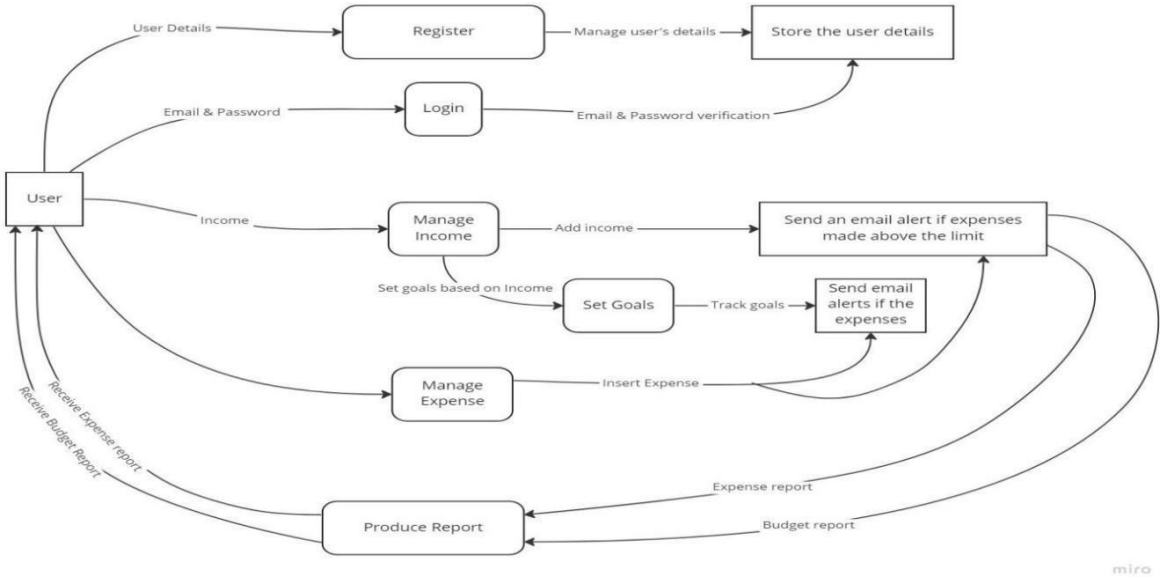
4.2.Non-Functionalrequirements:

FRNo.	Non-FunctionalRequirement	Description
NFR-1	**Usability**	Byusingthisapplication,theusercankeeptrackof theirexpensesandcanensurethatuser’smoneyis usedwisely.
NFR-2	**Security**	Maintainuserpersonaldetailsinaencrypted mannerbyusingdatasecurityalgorithms.

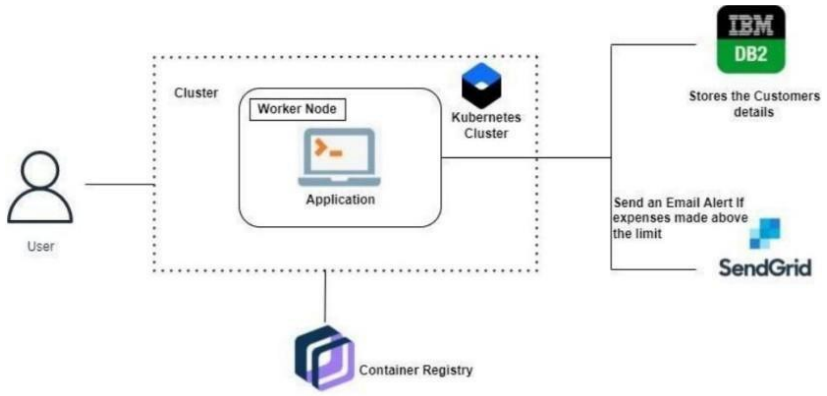
NFR-3	**Reliability**	It will maintain a proper tracking of day-to-day Expenses In an efficient manner.
NFR-4	**Performance**	By enter our in coming and departing cash,and the software can help you keep and monitor it with at most quality and security with high performance.
NFR-5	**Availability**	Using charts and graphs may help you monitor your budgeting and assets.
NFR-6	**Scalability**	Rely on your budgeting app to track,stream line, And automate all there current expenses and Remind you on a timely basis.

5. PROJECT DESIGN:

5.1. Data Flow Diagrams:



5.2. Solution&TechnicalArchitecture:



5.3. User Stories:

UserType	Functional Requirement (Epic)	User Story Number	UserStory/ Task	Acceptance criteria	Priority	Release
Customer	Registration	USN-1	Asauser,I can registerfor theapplicationby	I can accessmyaccount /	High	Sprint-1

			entering myemail,password,and confirming mypassword.	**dashboard**		
		USN-2	**As a user, Ican log intotheapplication byentering email& password**	**Icanaccess The application**	**High**	**Sprint-1**
	Dashboard	**USN-3**	**As a user Ican enter myincome And expenditure details under**	**I can viewmy Daily expenses andtrack**	**High**	**Sprint-2**

			different categories.I can save mybills and setgoals.	**mygoals**		
		USN-4	**Receiveemail alert ifthe goal isachievedor theexpense exceedthe budget.**	**Mailreceived successfully**	**High**	**Sprint-3**
		USN-5	**Monthlyalerts can beset to paythebills**	**Alertsreceived successfully**	**Medium**	**Sprint-3**

		USN-6	User caninteract withwatson ifthey haveany queries.	Interaction implemented successfully	Medium	Sprint-4
Customer Care Executive		USN-7	As acustomer careexecutive, Ican Solvethelog in issuesand otherissues of theapplication	I canprovidesupport orsolution atanytime 24*7	Medium	Sprint-1
Administrator	Application	USN-8	As anadministrator I canupgrade or updatetheapplication.	I can fix thebugwhich arises forthe	Medium	Sprint-1

				Customers andusers application		

6. PROJECT PLANNING & SCHEDULING:

6.1. Sprint Planning & 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.	2	High	Sowndariyalakshmi
		USN-2	As a user, I will receive confirmation email once I have registeredfor the application	1	High	Gnana rethes
	Login	USN-3	As a user, I can log into the application by entering email &password	1	High	Gogularam
	Dashboard	USN-4	Logging in takes to the dashboard for the logged user.	`2	High	Nandhini
*Bug fixes, routine checks and improvisation by everyone in the team *Intended bugsonly*						
Sprint 2	Workspace	USN-1	Workspace for personal expense tracking	2	High	Gnana rethes
	Charts	USN-2	Creating various graphs and statistics of customer's data	1	Medium	Nandhini
	Connecting to IBM DB2	USN-3	Linking database with dashboard	2	High	Sowndariyalakshmi
		USN-4	Making dashboard interactive with JS	2	High	Gogularam

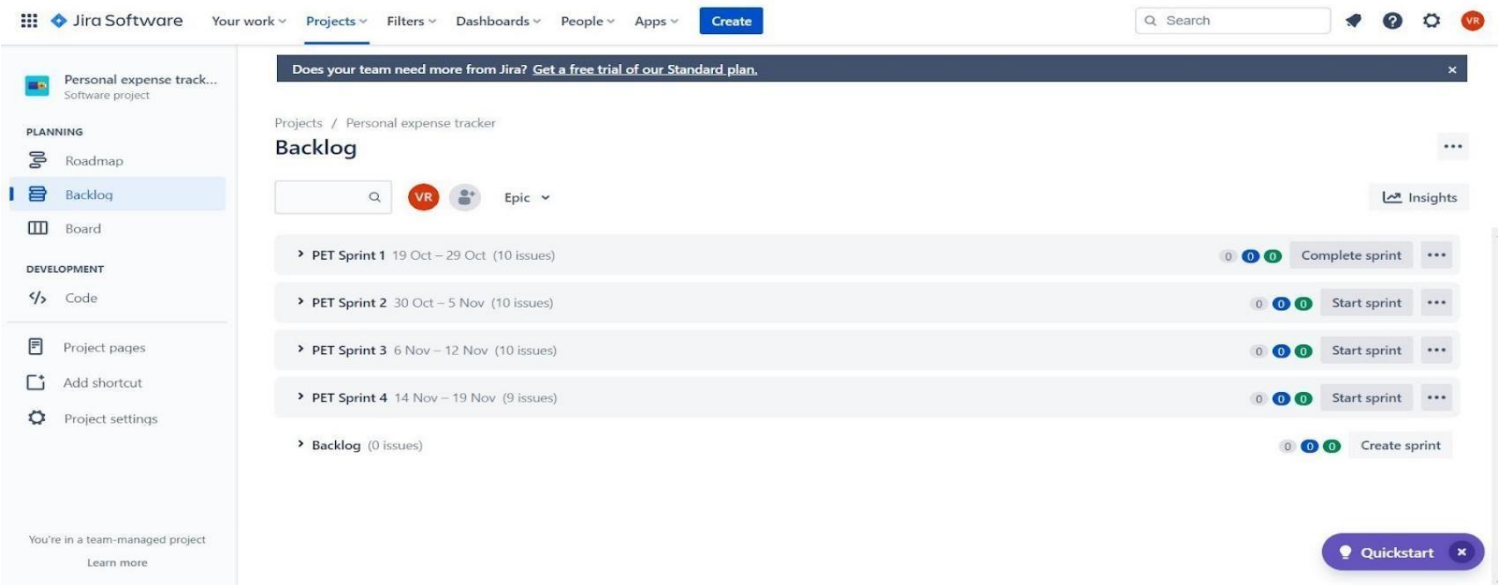
Sprint-3		USN-1	Wrapping up the server side works of frontend	1	Medium	Gnana rethes
	Watson Assistant	USN-2	Creating Chatbot for expense tracking and for clarifying user'squery	1	Medium	Gogularam
	SendGrid	USN -3	Using SendGrid to send mail to the user about their expenses	1	Low	Nandhini
		USN -4	Integrating both frontend and backend	2		Sowndariyalakshmi
*Bug fixes, routine checks and improvisation by everyone in the team *Intendedbugs only*						
Sprint-4	Docker	USN -1	Creating image of website using docker/	2	High	Nandhini
	Cloud Registry	USN -2	Uploading docker image to IBM Cloud registry	2	High	Sowndariyalakshmi
	Kubernetes	USN -3	Create container using the docker image and hosting the site	2	High	Gnana rethes
	Exposing	USN -4	Exposing IP/Ports for the site	2	High	Gogularam

6.2. Sprint Delivery Schedule:

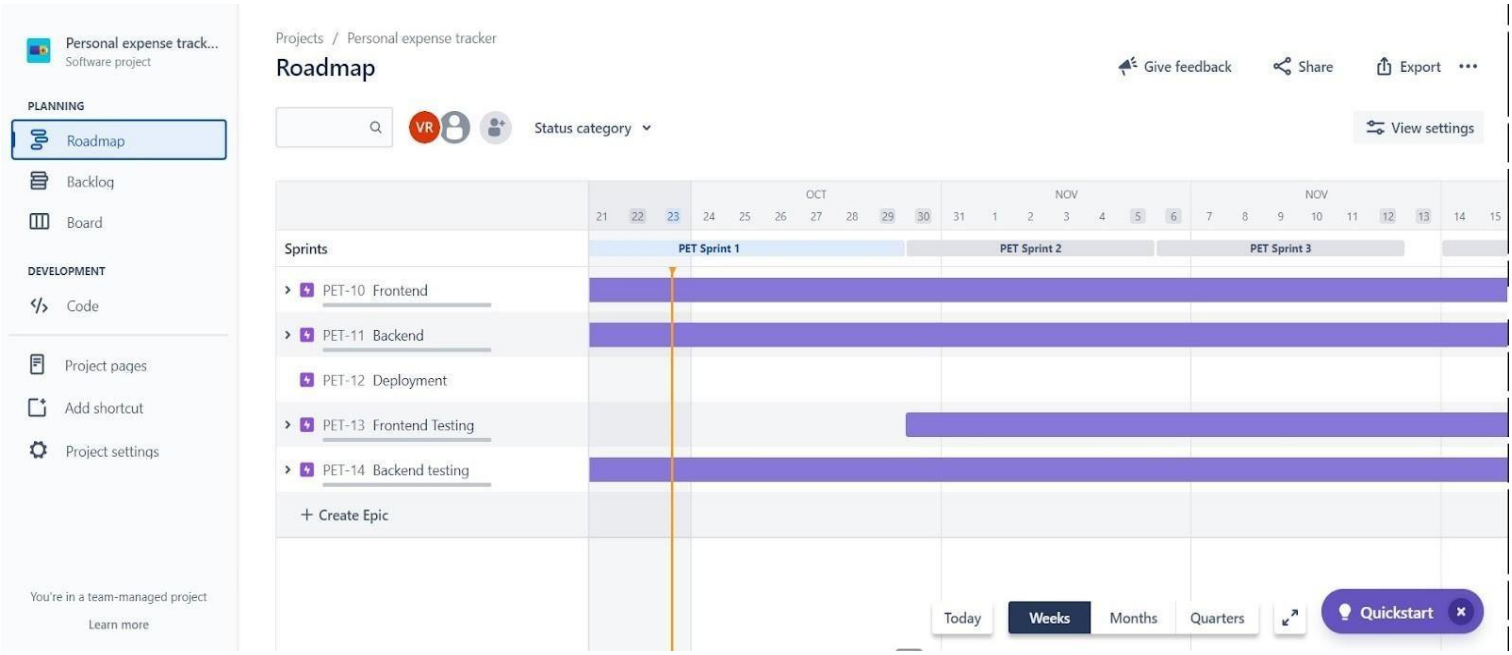
Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	05 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	12 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	19 Nov 2022

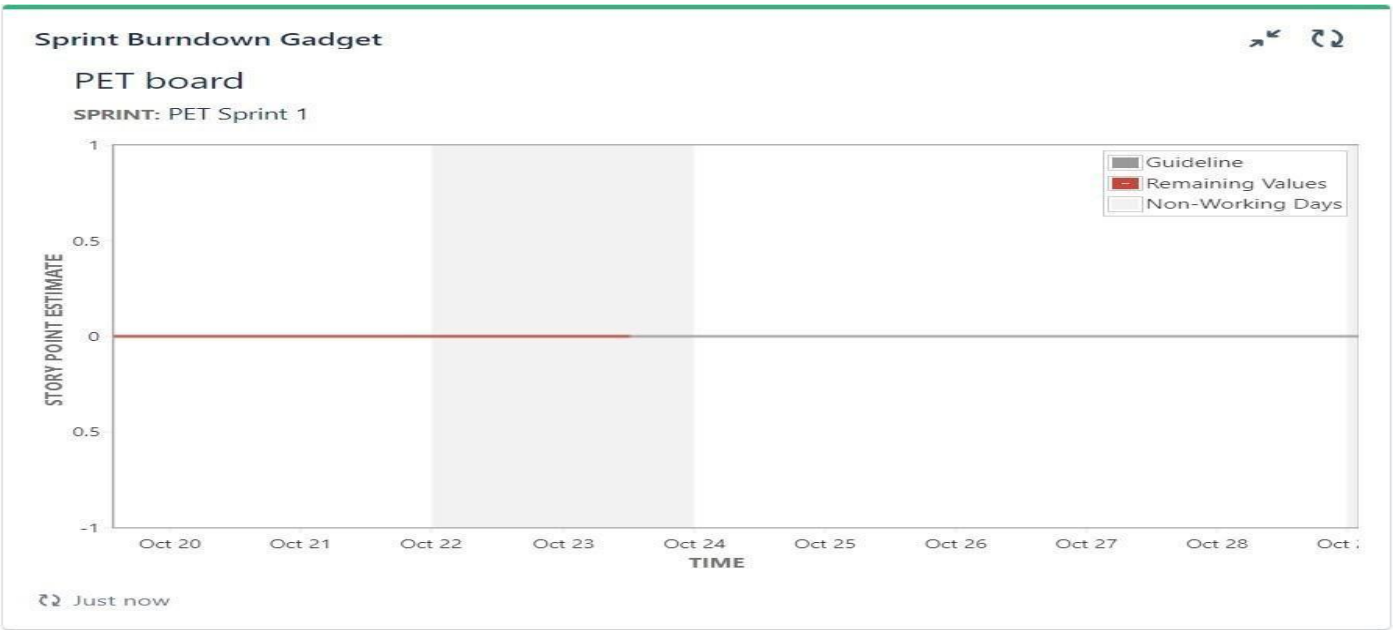
6.3. Reports from JIRA:

JIRA backlog tab that shows the Sprint wise schedule and progress of tasks



JIRA Roadmap that shows the completion of tasks with time.





7. CODING & SOLUTIONING(Explain the features added in the project along with code)

7.1. Feature1:

Adding transactions:

Users can add their transactions (expenses and income) to the records page and it will be displayed in the table. This can help them trackhow much they have earned and spent over time.

EXPLORER

index.js

SOURCE CODE

- Sprint 1
- Sprint 2
  - backend
  - node_modules
  - public
  - src
    - assets
    - Components
    - Pages
      - BillDashboard
      - BudgetDashboard
      - Login
      - RecordDashboard
        - index.js
        - recordDashboar...
      - Signup
    - App.css
    - App.js
    - index.js
    - reportWebVitals.js
    - store.js
  - .gitignore
  - package-lock.json
  - package.json
  - README.md
- Sprint 3
- Sprint 4

OUTLINE

TIMELINE

Sprint 2 > src > Pages > RecordDashboard > index.js

```
import React, { useEffect, useState } from 'react'
import Header from '../../Components/Header'
import SideBar from '../../Components/SideBar'
import { Box, Button, Checkbox, Container, Dialog, DialogActions, DialogContent, DialogContentText, DialogTitle, Divider, FormControlLabel, Paper, Table, Table
import emptyImg from '../../assets/empty_item.svg'
import './recordDashboard.css'
import BudgetCard from '../../Components/BudgetCard'
import { recordStore } from '../../store'
function RecordDialog(props) {
    const { onClose, open } = props;

    const handleClose = () => {
        onClose();
    };
    const handleChange = (event) => {
        props.setExpense({ ...props.expense, "gain": event.target.checked })
    };

    return (
        <Dialog onClose={handleClose} open={open}>
            <DialogTitle>Add expense</DialogTitle>
            <DialogContent>
                <TextField
                    autoFocus
                    margin="dense"
                    required
                    label="Amount"
                    type="number"
                    onChange={(e) => props.setExpense({ ...props.expense, "amount": e.target.value })}
                    fullWidth
                    variant="outlined"
                />
                <TextField
                    autoFocus
                    margin="dense"
                    required
                    label="Category"
                    type="text"
                    onChange={(e) => props.setExpense({ ...props.expense, "category": e.target.value })}
                    fullWidth
```

EXPLORER

index.js

SOURCE CODE

- Sprint 1
- Sprint 2
  - backend
  - node_modules
  - public
  - src
    - assets
    - Components
    - Pages
      - BillDashboard
      - BudgetDashboard
      - Login
      - RecordDashboard
        - index.js
        - recordDashboar...
      - Signup
    - App.css
    - App.js
    - index.js
    - reportWebVitals.js
    - store.js
  - .gitignore
  - package-lock.json
  - package.json
  - README.md
- Sprint 3
- Sprint 4

OUTLINE

TIMELINE

Sprint 2 > src > Pages > RecordDashboard > index.js > ...

```
                    label="Category"
                    type="text"
                    onChange={(e) => props.setExpense({ ...props.expense, "category": e.target.value })}
                    fullWidth
                    variant="outlined"
                />
                <FormControlLabel control={<Checkbox defaultChecked value={props?.expense?.gain} onChange={handleChange} />} label="is Expense" />
            </DialogContent>
            <DialogActions>
                <Button onClick={handleClose}>Done</Button>
            </DialogActions>
        </Dialog>
    );
}

const RecordDashboard = () => {
    const [hasExpense, setHasExpense] = useState(false)
    const [expense, setExpense] = useState(null);
    const records = recordStore.useState(s => s.records);
    const [open, setOpen] = useState(false);
    const handleClickOpen = () => {
        setOpen(true);
    };
    const getRecords = async () => {
        let token = localStorage.getItem('token');
        await fetch('http://localhost:5000/records', {
            method: 'GET',
            headers: new Headers({
                "x-access-token": token
            })
        }).then(res => res.json().then(data => {
            console.log(data)
            recordStore.update(s => {
                s.records = data.records
            })
        }))
    }
    useEffect(() => {
        getRecords()
    }, [records])
```

EXPLORER

index.js

SOURCE CODE

- Sprint 1
- Sprint 2
  - backend
  - node_modules
  - public
  - src
    - assets
    - Components
    - Pages
      - BillDashboard
      - BudgetDashboard
      - Login
      - RecordDashboard
        - index.js
        - recordDashboar...
      - Signup
    - App.css
    - App.js
    - index.js
    - reportWebVitals.js
    - store.js
  - .gitignore
  - package-lock.json
  - package.json
  - README.md
- Sprint 3
- Sprint 4

OUTLINE

TIMELINE

Sprint 2 > src > Pages > RecordDashboard > index.js > ...

```
    useEffect(() => {
        getRecords()
    }, [records])
    const formData = new FormData();
    const handleClose = (value) => {
        if (expense) {
            setHasExpense(true)
            formData.append("amount", expense.amount)
            formData.append("category", expense.category)
            formData.append("gain", expense.gain)
            let token = localStorage.getItem('token')
            fetch('http://localhost:5000/records', {
                method: 'POST',
                body: formData,
                headers: new Headers({
                    "x-access-token": token
                })
            }).then(res => res.json().then(data => {
                console.log(data)
                getRecords()
            }))
        }
        setOpen(false);
    };
    return (
        <>
            <SideBar />
            <div className='dashBoardContainer'>
                <Header />
                <div className="record__body">
                    {records.length > 0 ? (
                        <div className="record__body__container">
                            {/* <Paper elevation={5} className="record__ChartCont">
                                <Typography variant="h6" color="text.secondary" gutterBottom>
                                    Chart
                                </Typography>
                            </Paper> */}
                            <Paper elevation={5} style={{ padding: "20px" }} className="record__TableCont">
                                <Box sx={{ display: "flex", justifyContent: "space-between", alignItems: "center" }} >
```

EXPLORER

index.js

SOURCE CODE

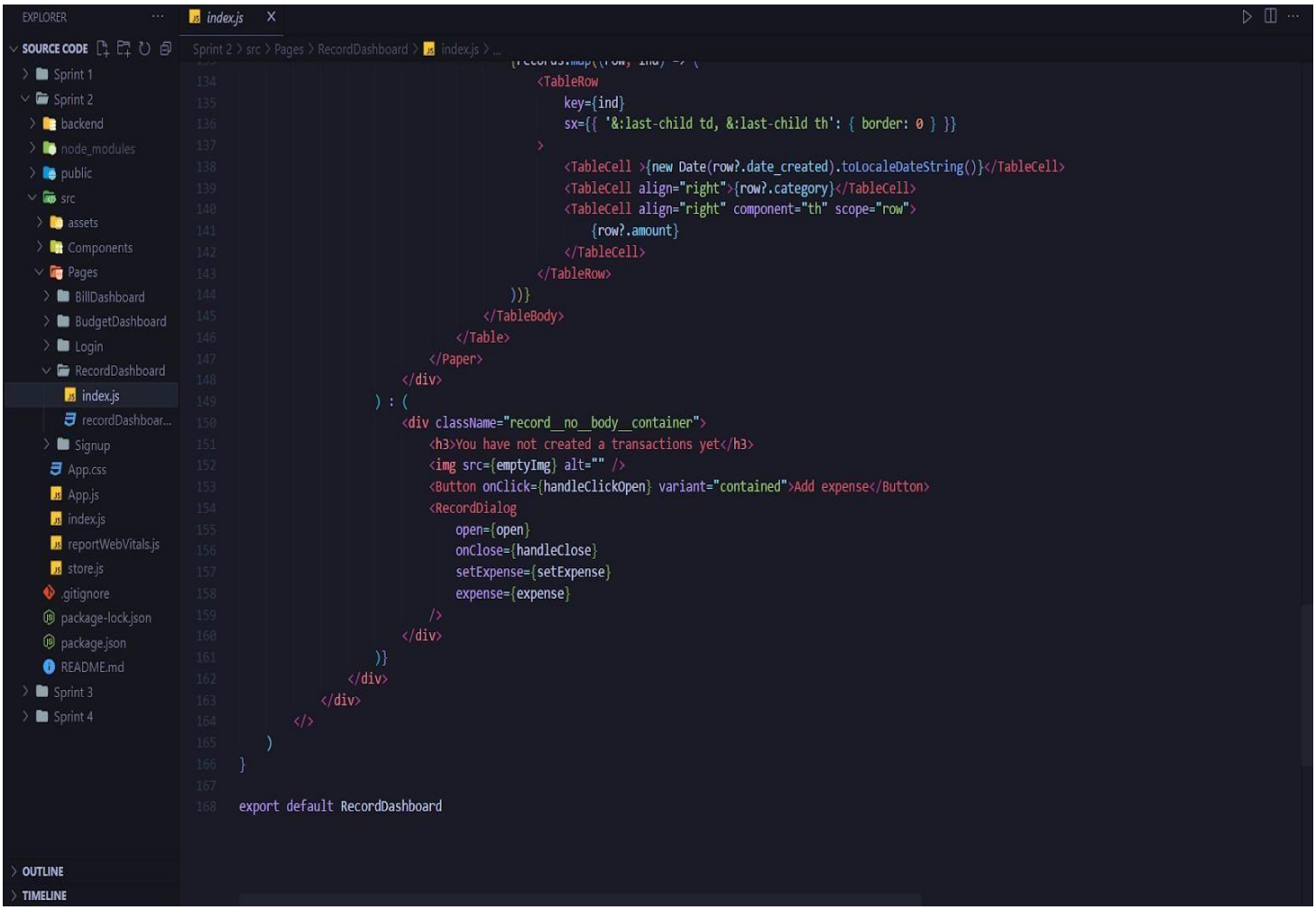
Sprint 2 > src > Pages > RecordDashboard > index.js > ...

- Sprint 1
- Sprint 2
  - backend
  - node_modules
  - public
  - src
    - assets
    - Components
    - Pages
      - BillDashboard
      - BudgetDashboard
      - Login
      - RecordDashboard
        - index.js
        - recordDashboar...
      - Signup
    - App.css
    - App.js
    - index.js
    - reportWebVitals.js
    - store.js
  - .gitignore
  - package-lock.json
  - package.json
  - README.md
- Sprint 3
- Sprint 4

OUTLINE

TIMELINE

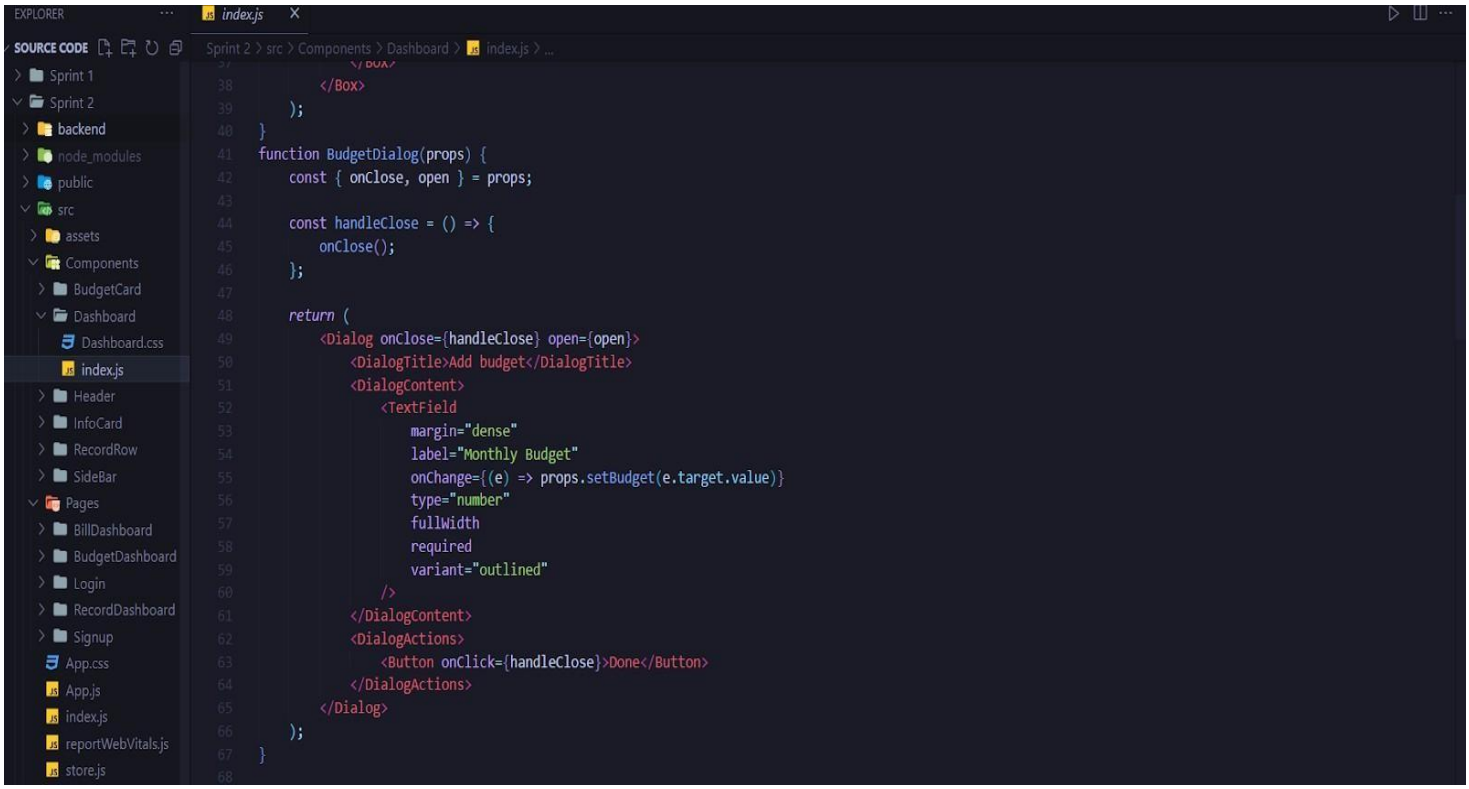
```
            <SideBar />
            <div className='dashBoardContainer'>
                <Header />
                <div className="record__body">
                    {records.length > 0 ? (
                        <div className="record__body__container">
                            {/* <Paper elevation={5} className="record__ChartCont">
                                <Typography variant="h6" color="text.secondary" gutterBottom>
                                    Chart
                                </Typography>
                            </Paper> */}
                            <Paper elevation={5} style={{ padding: "20px" }} className="record__TableCont">
                                <Box sx={{ display: "flex", justifyContent: "space-between", alignItems: "center" }} >
                                    <Typography variant="h6" color="text.secondary" gutterBottom>
                                        Records
                                    </Typography>
                                    <Button onClick={handleClickOpen} variant="contained">Add expense</Button>
                                    <RecordDialog
                                        open={open}
                                        onClose={handleClose}
                                        setExpense={setExpense}
                                        expense={expense}
                                    />
                                </Box>
                                <Table sx={{ minWidth: 650 }} aria-label="simple table">
                                    <TableHead>
                                        <TableRow>
                                            <TableCell>Date</TableCell>
                                            <TableCell align="right">Category</TableCell>
                                            <TableCell align="right">Amount</TableCell>
                                        </TableRow>
                                    </TableHead>
                                    <TableBody>
                                        {records.map((row, ind) => (
                                            <TableRow
                                                key={ind}
                                                sx={{ '&:last-child td, &:last-child th': { border: 0 } }}
                                            >
                                                <TableCell >{new Date(row?.date_created).toLocaleDateString()}</TableCell>
                                                <TableCell align="right">{row?.category}</TableCell>
```

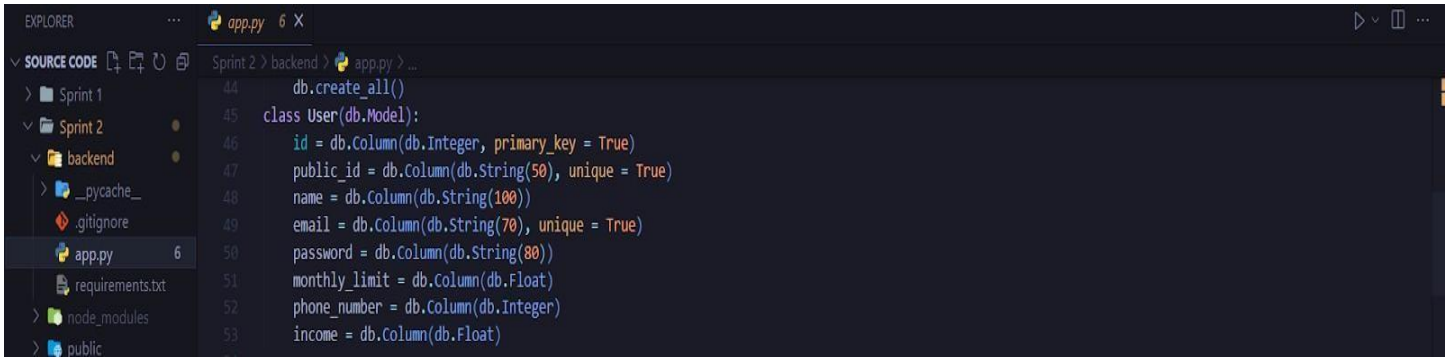


7.1.Feature2:

Tracking Budgets:

Users can create their budget limits and our app sends you and mail alert when the total expenses exceed the limit.





7.3. Database schema (If applicable):

3 separate schemas for Bills, records and Users were created.

EXPLORER

app.py 5

SOURCE CODE

- Sprint 1
- Sprint 2
  - backend
    - __pycache__
    - .gitignore
    - app.py
    - requirements.txt
  - node_modules
  - public
  - src
  - .gitignore
  - package-lock.json
  - package.json
  - README.md
- Sprint 3
- Sprint 4
  - backend
    - app.py 5
    - requirements.txt
  - README.md

OUTLINE

TIMELINE

Sprint 4 > backend > app.py > ...

```python
with app.app_context():
    db.create_all()
class User(db.Model):
    id = db.Column(db.Integer, primary_key = True)
    public_id = db.Column(db.String(50), unique = True)
    name = db.Column(db.String(100))
    email = db.Column(db.String(70), unique = True)
    password = db.Column(db.String(80))
    monthly_limit = db.Column(db.Float)
    phone_number = db.Column(db.Integer)
    income = db.Column(db.Float)

class Record(db.Model):
    id = db.Column(db.Integer,primary_key = True)
    user = db.Column(db.String(50))
    category = db.Column(db.String(50))
    date_created = db.Column(db.DateTime(timezone=True),default=datetime.utcnow)
    amount = db.Column(db.Float)
    gain = db.Column(db.Boolean)
    @property
    def serialize(self):
       return {
           'id'         : self.id,
           'user':self.user,
           'category':self.category,
           'date_created':self.date_created,
           'amount':self.amount,
           'gain':self.gain
           # This is an example how to deal with Many2Many relations
       }

class Bills(db.Model):
    id = db.Column(db.Integer, primary_key = True)
    user = db.Column(db.String(50))
    name = db.Column(db.String(50))
    due_date = db.Column(db.Date)
    amount = db.Column(db.Float)
    date_created = db.Column(db.DateTime(timezone=True),default=datetime.utcnow)
    @property
```

EXPLORER

app.py 5

SOURCE CODE

- Sprint 1
- Sprint 2
  - backend
    - __pycache__
    - .gitignore
    - app.py
    - requirements.txt
  - node_modules
  - public
  - src
  - .gitignore
  - package-lock.json
  - package.json
  - README.md
- Sprint 3
- Sprint 4
  - backend
    - app.py 5
    - requirements.txt
  - README.md

Sprint 4 > backend > app.py > ...

```python
    @property
    def serialize(self):
       return {
           'id'         : self.id,
           'user':self.user,
           'category':self.category,
           'date_created':self.date_created,
           'amount':self.amount,
           'gain':self.gain
           # This is an example how to deal with Many2Many relations
       }

class Bills(db.Model):
    id = db.Column(db.Integer, primary_key = True)
    user = db.Column(db.String(50))
    name = db.Column(db.String(50))
    due_date = db.Column(db.Date)
    amount = db.Column(db.Float)
    date_created = db.Column(db.DateTime(timezone=True),default=datetime.utcnow)
    @property
    def serialize(self):
       return {
           'id'         : self.id,
           'user':self.user,
           'name':self.name,
           'date_created':self.date_created,
           'due_date':self.due_date,
           'amount':self.amount,
           # This is an example how to deal with Many2Many relations
       }
# decorator for verifying the JWT
```

8.Testing

8.1 User Acceptance testing:

This report shows the number of resolved or closed bugs at each severity level, and how they were resolved

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	1	0	0	0	1
Duplicate	1	0	0	0	1
External	3	1	0	0	4
Fixed	4	1	0	0	5
Not Reproduced	0	0	0	0	1
Skipped	0	0	0	0	0
Won't Fix	0	0	0	0	0
Totals	9	2	0	0	11

This report shows the number of test cases that have passed, failed, and untested

Section	Total Cases	Not Tested	Fail	Pass
Print Engine	0	0	0	0
Client Application	5	0	0	5
Security	0	0	0	0
Outsource Shipping	0	0	0	0
Exception Reporting	5	0	0	5
Final Report Output	0	0	0	0
Version Control	0	0	0	0

9.Results

9.1Performance metrics:

	NFT - Risk Assessment						
Scope/feature	Functional Changes	Hardware Changes	Software Changes	Impact of Downtime	Load/Volume Changes	Risk Score	Justification
New	Low	No Changes	Moderate	Yes, 2hrs	>10 to 30%	GREEN	

NFT - Detailed Test Plan				
S.No	Project Overview	NFT Test approach	Assumptions/Dependencies/Risks	Approvals/SignOff
1	Login Page	1) Open the Personal Expense Tracker Application 2) Login with user Credentials	No Risks	N/A
2	Signup Page	1) Open the Personal Expense Tracker Application 2) Enter the Details and Create a new User	No Risks	N/A
3	Records Page	1) Log in to Personal Expense Tracker Application 2) Enter all the pesonal details and expenses and mark it as expense or income	No Risks	N/A
4	Dashboard	1) Log in to Personal Expense Tracker Application 2) View the Analytics	No Risks	N/A
5	Bills Page	1) Log in to Personal Expense Tracker Application 2) Bills can be added.	No Risks	N/A
5	Email Acknowledgement	1) Mails are Sent to the Registered user if expenses>budget	No Risks	N/A

	End Of Test Report					
NFT Test approach	NFR - Met	Test Outcome	GO/NO-GO decision	Recommendations	Identified Defects (Detected/Closed/Open)	Approvals/SignOff
1) Log in to Personal Expense Tracker Application 2) Test for all Testcases 3) Log out to Personal Expense Tracker Application	YES	Test Passed	GO/NO-GO decision	N/A	None	N/A

10.Advantages and Disadvantages

Advantages

• This system helps users to reduce their expenses.

• Easy to use.

• You’llhavebetterinsightintoyourspendinghabits.

• Provides a better overview and comprehensive analysis.

• Limit your spending.

Disadvantages

• Need to spend specified time to enter data.

• On some occasions it is not mandatory to restrict expenses.

11. Conclusion

PersonalExpenseTrackerisawebapplication.Wecreatedthisapplicationsothata user can accurately calculatehis/her daily cost. After makingthis application we assure thatthis application will help its users to manage the cost of their daily expenditure. It will guidethem and make them aware about their daily expenses. It will prove to be helpful for the peoplewho are frustrated with their daily budget management, irritated because of the amount of expenses and wish to manage money and to preserve the record of their daily cost which may beuseful to change their way of spending money. In short, this application will help its users toovercomethe wastage of money.

12. Futurescope

Now in our web application we covered almost all features but in the future wewill add some more futures. The features are below:

• Multiple account support.

• Givinganalertmessagetotheuser'smobilenumber.

• Includecurrencyconverter.

• Reports are created in categories.

• Giving users the ability to export the data in all available formats like pdf, excel etc…

13. Appendix

13.1 Sourcecode:

App.py (Flask file):

```
fromflaskimportFlask,request,jsonify,make_response,current_appfromflask_
sqlalchemyimportSQLAlchemy
importuuid
from sqlalchemy import
extractfromflask_corsimportCORS#importibm_db
#importibm_db_sa fromwerkzeug.securityimportgenerate_password_hash,check_password_hash#imp
ortsforPyJWTauthentication
importjwt fromdatetimeimportdatetime,timedeltafrom
functoolsimportwraps fromwaitressimportserve
#fromibm_db_alembic.ibm_dbimportIbmDbImplimport os
from sendgrid import
SendGridAPIClientfromsendgrid.helpers.
mailimportMail

# creates Flask objectapp =
Flask(name)CORS(app)

#configuration1
#NEVERHARDCODEYOURCONFIGURATIONINYOU
RCODE
#withinthisblock,current_apppointstoapp.#INSTEA DCREATEA.envFILEANDSTOREINIT
app.config['SECRET_KEY'] = ''#databasename
app.config['CORS_HEADERS']='Content-Type'
#app.config['SQLALCHEMY_DATABASE_URI']='sqlite:///Database.db'app.
config['SQLALCHEMY_DATABASE_URI']=''
```

```
app.config['SQLALCHEMY_TRACK_MODIFICATIONS']=False
# creates SQLALCHEMY
objectdb=SQLAlchemy(app)#DatabaseORMs


default_info =
        {'name':
        '','limit':5000,
        'phone_number':
        0,'currency':        ₹
        '₹','income':
        0,'category':'misc'
}


#withapp.app_context():

#

            db.create_all()

classUser(db.Model):
        id = db.Column(db.Integer, primary_key=True)public_id=db.Column(db.String(50)
        ,unique=True)name=db.Column(db.String(100)) email =
        db.Column(db.String(70), unique=True)password =
        db.Column(db.String(110))monthly_limit =
        db.Column(db.Float)phone_number=db.Column(db.
        String(20))income=db.Column(db.Float)


classRecord(db.Model):
        id = db.Column(db.Integer, primary_key=True)user=db.Column(db.String(50)
        )
        category =
```

```
db.Column(db.String(50))date_created= db.Column(db.DateTime(
        timezone=True),default=datetime.utcnow)
```

```
    amount = db.Column(db.Float)gain=db.C
    olumn(db.Boolean)

    @propertydef
        serialize(self):
        return{
            'id':
            self.id,'user':self.user,
            'category':
            self.category,'date_created':
            self.date_created,'amount':self.amount,
            'gain':self.gain #ThisisanexamplehowtodealwithMany2Manyrelations
        }


classBills(db.Model):
    id = db.Column(db.Integer, primary_key=True)user=db.Column(db.String(50)
    )
    name = db.Column(db.String(50))due_dat e
    = db.Column(db.Date)amount=db.Col
    umn(db.Float)
    date_created = db.Column(db.DateTime(timezone=True),def
        ault=datetime.utcnow)

    @propertydef
        serialize(self):return{
            'id': self.id,'user':
```

```
        self.user,'name':s
        elf.name, 'date_created':
        self.date_created,'due_date': self.due_date,'amount':self.amount,
        #ThisisanexamplehowtodealwithMany2Manyrelations
}
```

```
#decoratorforverifyingtheJWT


 def
        token_required(f):
        @wraps(f)
        def decorated(*args,
                **kwargs):token=None # jwt
                is passedin the
                requestheaderif'x-access- token'inrequest.headers:
                        token = request.headers['x-access-
                token']#return401iftokenisnotpassed ifnottoken:
                        returnjsonify({'message':'Tokenismissing!!'}),401

                  try:
                        #decodingthepayloadtofetchthestoreddetails#print("receivedtoken:",token)
                        #
                        print(app.config['SECRET_KEY'])data=jwt.decode(
                                token,app.config['SECRET_KEY'],algorithms=["HS256"])#p rint("data",data)
                        current_user=User.query\
                                .filter_by(public_id=data['public_id'])\
                                .first()
                except:
                        returnjsonify({ 'message':'Tokenisinvalid!!'
                        }),401
                #returnsthecurrentloggedinuserscontextotheroutesreturnf(cu
                rrent_user,*args,**kwargs)

        returndecorated

#UserDatabaseRoute #thisroutesendsbacklistofusers
```

```
@app.route('/user',
methods=['GET'])@token_required def
        get_all_users(current_user):#queryingthedatabase
        # for all the entries in
        itusers=User.query.all()#
        converting the query
        objects#tolistofjsons output=[]
        foruserinusers:
                # appending the user datajson# to
                the response listoutput.append({
                        'public_id':
                        user.public_id,'name':user.name,
                        'email':user.email
                })

        res = jsonify({'users': output})res.headers['Access-Control-
        Allow-
        Origin'] = '*'returnres


#defuser_has_exceeded_send_email(current_user):


@app.route('/getinfo',
methods=['GET'])@token_required def
        get_info(current_user):
        output={}

        output['public_id'] = current_user.public_idoutput['name'] =
        current_user.nameoutput['email']=current_use r.email
        output['monthly_limit']=current_user.monthly_limit
```

```
    output['phone_number']=current_user.phone_numberou
    tput['income']=current_user.income

    res = jsonify({'users': output})res.headers['Access-Control-
    Allow-
    Origin'] = '*'returnres
#routeforlogginguserin


@app.route('/login', methods=['POST'])deflogin():
    # creates dictionary of form
    dataauth=request.form

    ifnotauthornotauth.get('email')ornotauth.get('password'):#returns 401 if any email or/
        and password is missingreturnmake_response(
            'Could not
            verify',401,
            {'WWW-Authenticate':'Basicrealm="Loginrequired!!"'}
        )

    user=User.query\
        .filter_by(email=auth.get('email'))\
        .first()

    ifnotuser:
        # returns 401 if user does not
        existres=make_response(
            'Could not
            verify',401,
            {'WWW-Authenticate':'Basicrealm="Userdoesnotexist!!"'}
        )
        res.headers['Access-Control-Allow-Origin'] ='*'returnres
```

```
    ifcheck_password_hash(user.password,auth.get('password')):#g eneratestheJWTToken
        token=jwt.encode({'public_id':us
            er.public_id,
            'exp':datetime.utcnow()+timedelta(minutes=24*60*10)
        },app.config['SECRET_KEY'],algorithm="HS256")

        res=make_response(jsonify({'token':token}),201)res. headers['Access-
        Control-Allow-Origin'] = '*'returnres
    # returns 403 if password is
    wrongres=make_response(
        'Could not
        verify',403,
        {'WWW-Authenticate':'Basicrealm="WrongPassword!!"'}
    )
    res.headers['Access-Control-Allow-Origin'] ='*'returnres


#signuproute @app.route('/signup',
methods=['POST'])defsignup():
    # creates a dictionary of the form
    datadata=request.form

    #getsname,emailandpassword name,email=data.get('name'),data.get('email')pass
    word=data.get('password')
    income = data.get('income') if
        data.get('income')elsedefault_info['i ncome']
    monthly_limit=data.get('monthly_limit')ifdata.get('monthly_limit')elsedefault_info['limit']
    phone_number = data.get('phone_number') if
        data.get('phone_number')elsedefault_info['phone_n umber']
    #checkingforexistinguser
```

```
        user=User.query\
            .filter_by(email=email)\
            .first()i
        fnotuser:
            # database ORM
            objectuser=User(
                public_id=str(uuid.uuid4()),
                name=name,
                email=email,password=generate_password_has
                h(password),income=income,monthly_limit=mo
                nthly_limit,phone_number=phone_number
            )
            # insert userdb.session.add(u
            ser)db.session.commit()

            res=make_response('Successfullyregistered.',201)res.headers['Access-
            Control-Allow-Origin']='*'
        else:
            #returns202ifuseralreadyexists
            res=make_response('Useralreadyexists.PleaseLogin.',202)res.head ers['Access-Control-
            Allow-Origin']='*'
        returnres


@app.route('/bills', methods=['GET'])@token_required
defget_bills(current_user):
        bills=Bills.query.filter_by(user=current_user.public_id).all()if billsisNone:
            bills=[] res=make_response(jsonify({'bills':[i.serializeforiinbills]}),
201)
        res.headers['Access-Control-Allow-Origin']='*'
```

```
    returnres


@app.route('/records',
methods=['GET'])@token_required
defget_record(current_user):
    records=Record.query.filter_by(user=current_user.public_id).all()ifrecordsisNone:
        records={}res=ma
    ke_response(
        jsonify({'records':[i.serializeforiinrecords]}),201)res.headers['Access-Control-
    Allow-Origin']='*'
    returnres


@app.route('/records',
methods=['POST'])@token_requireddef
    put_record(current_user):
    form=request.form
    ifnotform:
        res=make_response('couldnotaddrecordnodatareceived',401)res.heade rs['Access-Control-
        Allow-Origin']='*'
        returnres
    if not form.get('category') or (form.get('gain') is None) ornotform.get('amount'):
        res=make_response( 'couldnotaddrecordnoenoughdatareceived',401)res.head
        ers['Access-Control-Allow-Origin']='*'return
    resrecord=Record(
        user=current_user.public_id,category=form.get('category')ifform.get(
            'category') else default_info['category'],amount=form.get('amou
        nt'),gain=form.get('gain')=="True"
    )
```

```
db.session.add(record)db.session.commit()

dt = datetime.utcnow()recor
d_this_month=
Record.query.filter_by(user=current_user.public_id).filter(db.extract('year',
Record.date_created)==dt.year,db.extract('month',
Record.date_created)==dt.month)
current_month_spending=sum(
[-1*i.amount if i.gain else i.amount for iinrecord_this_month.all()])
ifcurrent_month_spending>=current_user.monthly_limit:message=Mail(
from_email='210419104166@smartinternz.com',
to_emails=current_user.email,
subject='YourMonthlyexpenseshaveexceededyourtarget
budget.',
html_content=f'''
<strong>Hey{current_user.name}!</strong><br>
<imgsrc="https://image.shutterstock.com/image-vector/white-
coupon-banner-word-over-260nw-2213547155.jpg"alt="overbudget!"/>
YourMonthlyexpenseshaveexceededyourtargetbudget.<br>Kindlyvisit the Expense
application for more insights.<br>Visit:expenso


Thank you! keep
Tracking!<br>AdiosAmigos.!''')try:
sg=SendGridAPIClient(os.environ.get('SENDGRID_API_KEY'))response=sg.send(message)
#
print(response.status_code)#p
rint(response.body)
#
print(response.headers)except Exceptionase:
```

```
                    print("emailerror",e)

            res=make_response("sucessfullyaddedrecord",201)res.headers['Access-
            Control-Allow-Origin'] = '*'returnres


        @app.route('/bills',
        methods=['POST'])@token_requireddef
            put_bills(current_user):
            form=request.form
            ifnotform:
                res=make_response('couldnotaddrecordnodatareceived',401)res.heade rs['Access-Control-
                Allow-Origin']='*'
                returnres
            if not form.get('amount') or not form.get('due_date') or
notform.get('amount')ornotform.get('bill_name'):
                res=make_response( 'couldnotaddrecordnoenoughdatareceived',401)res.head
                ers['Access-Control-Allow-Origin']='*'return
            resbills=Bills(
                user=current_user.public_id,amount=form.get('amount'),
                due_date=datetime.strptime(form.get('due_date'),"%Y-%m-

%d").date(),            name=form.get('bill_name')

             )

            db.session.add(bills) db.session.commit()
            res=make_response("sucessfullyaddedbill",201)res.headers['Access-
            Control-Allow-Origin'] = '*'returnres
```

```
@app.route('/dashboard',
methods=['GET'])@token_required def
    dashboard(current_user):
    dt = datetime.utcnow()record_
    this_month=
Record.query.filter_by(user=current_user.public_id).filter(db.extract('year',
        Record.date_created)==dt.year,db.extract('month',
Record.date_created)==dt.month)
    record_last_seven_days
=Record.query.filter_by(user=current_user.public_id).filter   (
        Record.date_created>(dt- timedelta(days=7))).all()last_week_spending=sum(
            [-1*i.amount if i.gain else i.amount for i
inrecord_last_seven_days])
        current_month_spending=sum(
            [-1*i.amount if i.gain else i.amount for i
inrecord_this_month.all()])
        if dt.month >
            1:last_month_spending
            =
Record.query.filter_by(user=current_user.public_id).filter(db.extract(
                'month',Record.date_created)==dt.month-1,
db.extract('year',Record.date_created)==dt.year).all()els e:
            last_month_spending
=Record.query.filter_by(user=current_user.public_id).filter(db.extract    (
                'month', Record.date_created) ==
12,db.extract('year',Record.date_created)==dt.year-1).all()
        income=current_user.incomeifcurrent_user.incomeisnot
Noneelse0
        balance = income - current_month_spendingby_category=
record_this_month.filter_by(gain=False).with_entities(Record.category,
db.func.sum(Record.amount)).group_by(Record.category).all()
```

```
category_list={}
```

```
                for x, y in
                    by_category:category
                    _list[x]=y
                response_obj =
make_response(jsonify({'last_week_spending':last_week_spending, 'expense_by_category':
category_list, 'income':income,'balance':balance,
                                                    'monthly_limit':current_user.mont
hly_limit, 'current_month_spending':current_month_spending,'last_month_spending':last
_month_spending}),201)
                response_obj.headers['Access-Control-Allow-
                Origin']='*'returnresponse_obj


            @app.route('/budget',
            methods=['POST'])@token_requireddef
                addbudget(current_user):
                form=request.form
                limit = form.get('budget')current_user.mon
                thly_limit = limitdb.session.add(current_user)d
                b.session.commit()
                res=make_response("sucessfullyaddedbudget",201)res.headers['Access-
                Control-Allow-Origin'] = '*'returnres


@app.before_first_request defcreate_tables():
    db.create_all()


ifname_______=="main":
    # setting debug to True enables hotreload#andalsoprovidesadebuggershell
    # if you hit an error while running the server#app.run(debug=True,host="0.0.0.0")serve
    (app,listen='*:5000')
```

13.2 Attachements:

Github link:

https://github.com/IBM-EPBL/IBM-Project-27235-1660051509

Video link:

IBM-Project-27235-1660051509\IBM-Project-27235-1660051509\Final Deliverables\Demo video.mkv