PPA Membership Maintenance System

Sprint Report 3

Capstone Computing Project 2

Group SD07

Semester 2, 2018

Curtin University – Department of Computing

Assignment Cover Sheet / Declaration of Originality

Complete this form if/as directed by your unit coordinator, lecturer or the assignment specification.

Last name:	Hingalagoda	Student ID:	19211804
Other name(s):	Bhanuka		
Unit name:	Capstone Computing Project 2	Unit ID:	ISAD3001
Lecturer / unit coordinator:	Dr.. Hannes Herrmann	Tutor:	Ms. Geethanjalie Wimalarathne
Date of submission:	14/09/2018	Which assignment?	Sprint Report 03

I declare that:

- The above information is complete and accurate.
- The work I am submitting is *entirely my own*, except where clearly indicated otherwise and correctly referenced.
- I have taken (and will continue to take) all reasonable steps to ensure my work is *not accessible* to any other students who may gain unfair advantage from it.
- I have *not previously submitted* this work for any other unit, whether at Curtin University or elsewhere, or for prior attempts at this unit, except where clearly indicated otherwise.

I understand that:

- Plagiarism and collusion are dishonest, and unfair to all other students.
- Detection of plagiarism and collusion may be done manually or by using tools (such as Turnitin).
- If I plagiarise or collude, I risk failing the unit with a grade of ANN (“Result Annulled due to Academic Misconduct”), which will remain permanently on my academic record. I also risk termination from my course and other penalties.
- Even with correct referencing, my submission will only be marked according to what I have done myself, specifically for this assessment. I cannot re-use the work of others, or my own previously submitted work, in order to fulfil the assessment requirements.
- It is my responsibility to ensure that my submission is complete, correct and not corrupted.

Signature: [illegible] Date of signature: 14/09/2018

(By submitting this form, you indicate that you agree with all the above text.)

Table of Contents

1. Introduction 2
1.1 Group Introduction 2
1.2 Project Introduction 2
2. Progress Update 3
2.1 Allocated Tasks for the Sprint 2 3
2.2 Planned Tasks for the Sprint 3 3
2.3 Difficulties 3
3. Task Break Down 5
3.1 Detailed Account Page Designing 5
3.2 Task 2 Detailed Account Page Implementation 7
3.3 Task 3 Detailed Account Page - Database Design 9
3.4 Task 4 Detailed Account Page - Insert Method Calls 10
3.5 Task 5 Detailed Account Page Update Method Calls 12
3.6 Task 6 Detailed Account Page Delete Method Calls 15
3.7 Task 7 Detailed Account Page Validation Methods 17
3.8 Task 8 Detailed Account Page Transaction Table Insert Methods 18
3.9 Task 9 Detailed Account Page Transaction Table Update Methods 20
3.10 Task 10 Detailed Account Page Transaction Table Delete Methods 23
3.11 Task 11 Sprint 4 Planning 25
3.12 Task 12 Discuss design issues Sprint 3 25
4. Development Methodology 26
4.1 Minutes 26
4.2 Burndown Chart 26
4.3 Sprint Retrospective 27
4.4 Task Summary 27
4.5 Time Management 27

1. Introduction

1.1 Group Introduction

All the members in our group have successfully completed the CCP1 in 2017. We enrolled for the CCP2 module in 2018 2nd semester, therefore this project is started in semester 2 of this year (2018). Because of that we have to done the main documentations and tasks of the project such as SRS, task allocation, initial requirement gathering etc. in semester 2. Each of the group members has to do a workload of 2 semesters within this semester, in order to complete the project successfully.

1.2 Project Introduction

PPA Membership Maintenance System is going to be used for membership management and some other important administration tasks such as event planning, donation collecting, accounts handling etc. of past pupil association of Sirimavo Bandaranaike Vidyalaya. Currently all these operations are manually performed by the committee. The main intention of the system is to automate most of those tasks and perform the semi-automated tasks easily and conveniently.

We use MEAN stack to develop this application, JIRA as the project management tool and bitbucket as the online repository. The application has main 4 parts as Membership Services, Accountings, Event Planning and Reporting. These four sections are interconnected with each other as per their functionalities.

2. Progress Update

Sprint 2: 1st September – 14th September

2.1 Allocated Tasks for the Sprint 3

#	Task ID	Task	Task Status	Hours Estimated
1	PPA-	Detailed Account Page Designing	Completed	1
2	PPA-	Detailed Account Page Implementation	Completed	4
3	PPA-	Detailed Account Page - Database Design	Completed	1
4	PPA-	Detailed Account Page - Insert Method Calls	Completed	2
5	PPA-	Detailed Account Page Update Method Calls	Completed	2
6	PPA-	Detailed Account Page Delete Method Calls	Completed	2
7	PPA-	Detailed Account Page Validation Methods	Completed	2
8	PPA-	Detailed Account Page Transaction Table Insert Methods	Completed	2
9	PPA-	Detailed Account Page Transaction Table Update Methods	Completed	2
10	PPA-	Detailed Account Page Transaction Table Delete Methods	Completed	2
11	PPA-	Sprint 4 Planning	Completed	1
12	PPA-	Discuss design issues Sprint 3	Completed	1
			Total Hours	**22**

2.2 Planned Tasks for the Sprint 4

#	Task ID	Task	Estimation (Hours)
1	PPA-	Detailed Account Page Transaction Table Search Methods	1
2	PPA-	Account Report Page GUI Designing	2
3	PPA-	Account Report Page GUI Implementation	3
4	PPA-	Account Report Page GUI Report Generation Algorithm	4
5	PPA-	Detailed Account Page Search & Filtering Method Calls	4
6	PPA-	Research about report generation	4
7	PPA-	Service Letter Generation GUI Designing & Implementation	3
8	PPA-	Service letter generation Report Designing	2
9	PPA-	Sprint 5 Planning	1
10	PPA-	Discuss design issues Sprint 4	1
		Total Hours	**25**

2.3 Difficulties

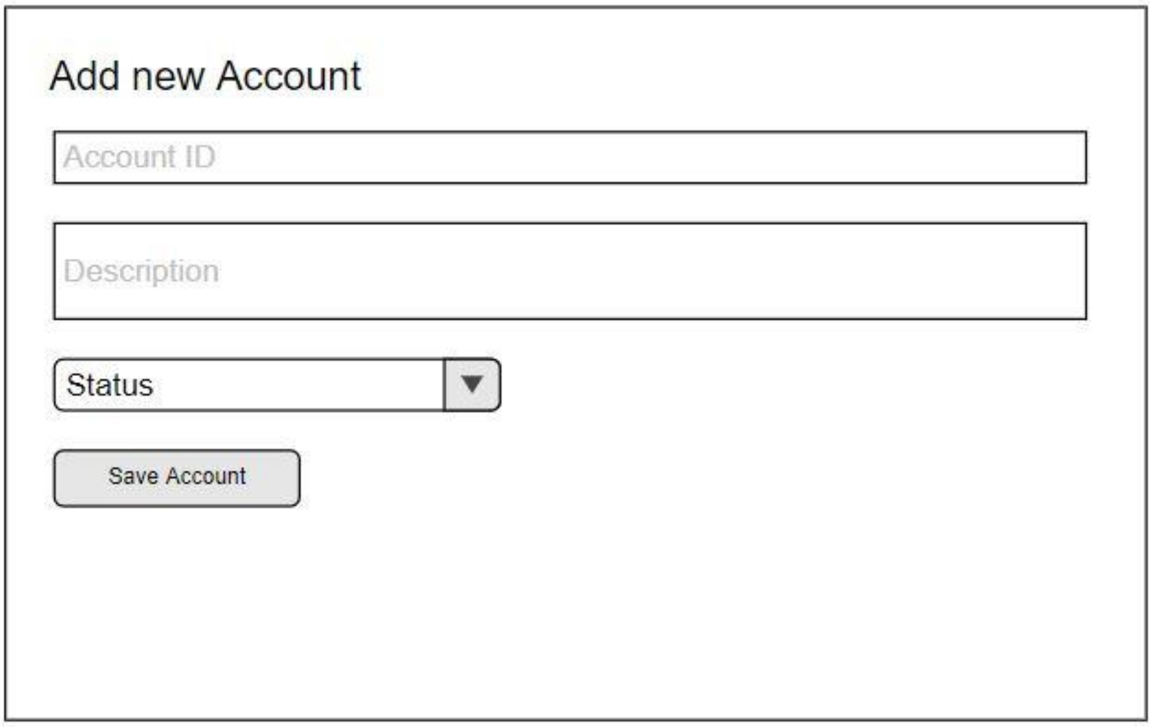
3. Task Break Down

3.1 Detailed Account Page Designing

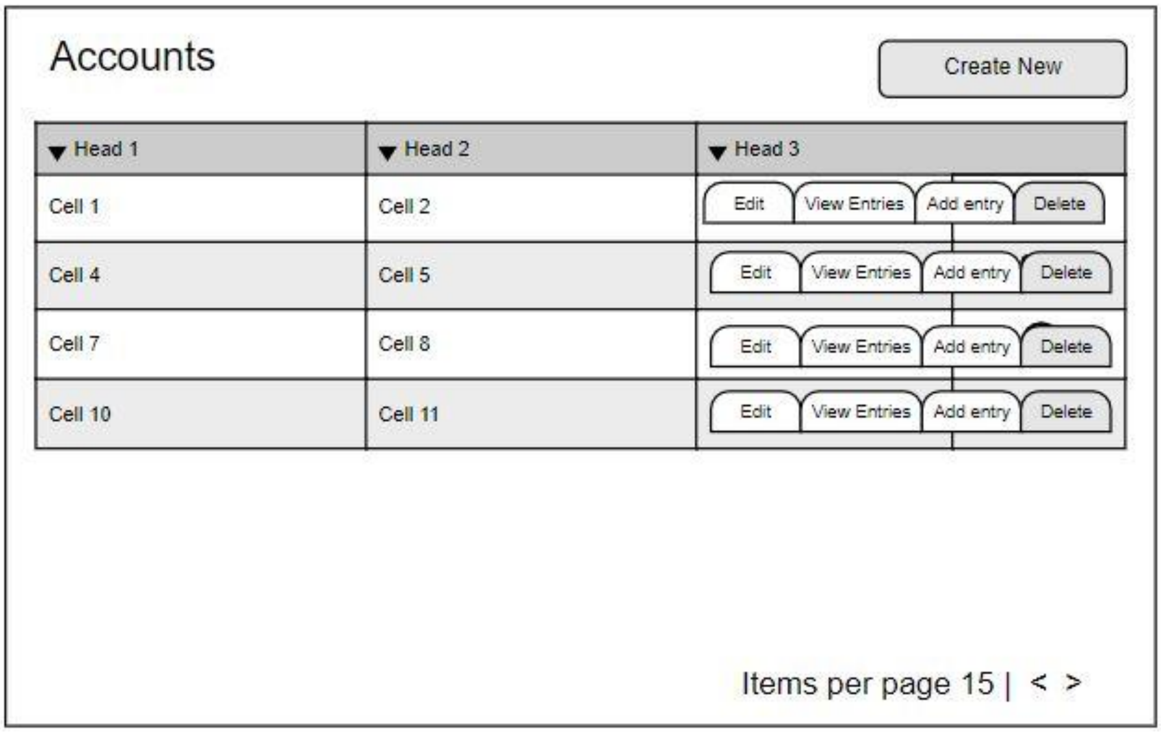
Estimate Time:	1 Hours
Actual Time:	1 Hours
Actual Time (this sprint):	1 Hours

Description

From this task I designed the pages needed in accounts for this sprint.



This window will be used to update as well



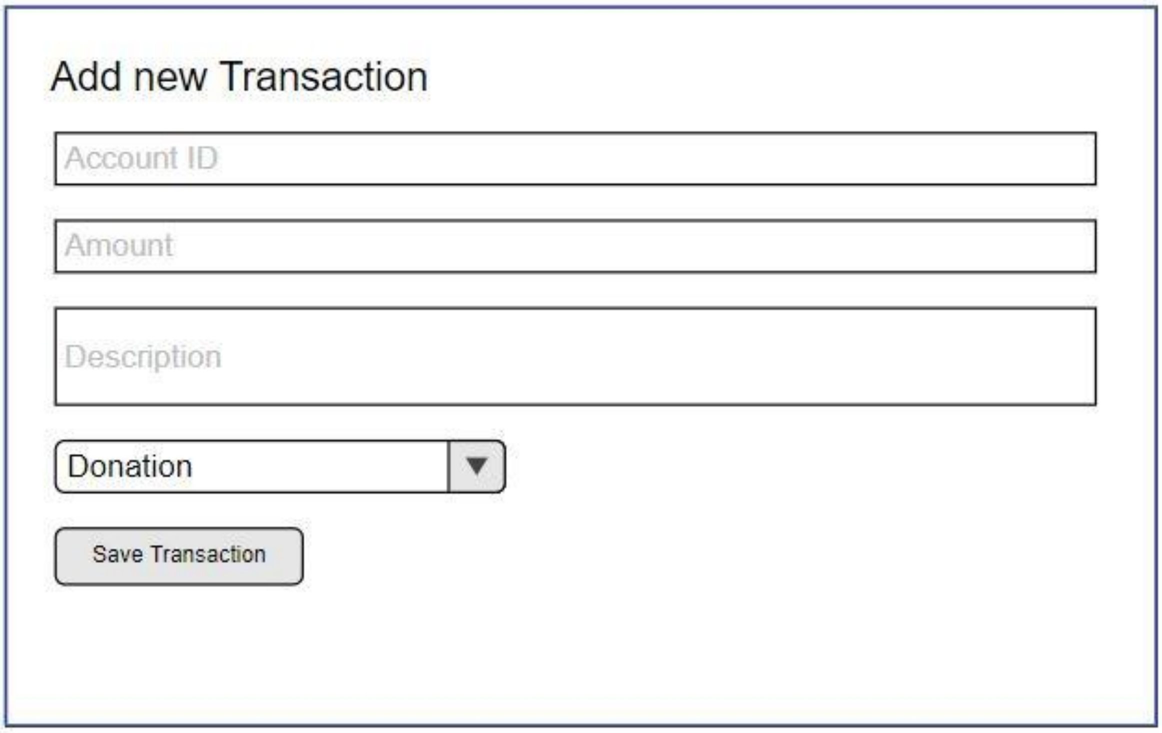
Accounts

Create New

▼ Head 1	▼ Head 2	▼ Head 3
Cell 1	Cell 2	Edit · View Entries · Add entry · Delete
Cell 4	Cell 5	Edit · View Entries · Add entry · Delete
Cell 7	Cell 8	Edit · View Entries · Add entry · Delete
Cell 10	Cell 11	Edit · View Entries · Add entry · Delete

Items per page 15 | < >

View All accounts



Add new Transaction

Account ID

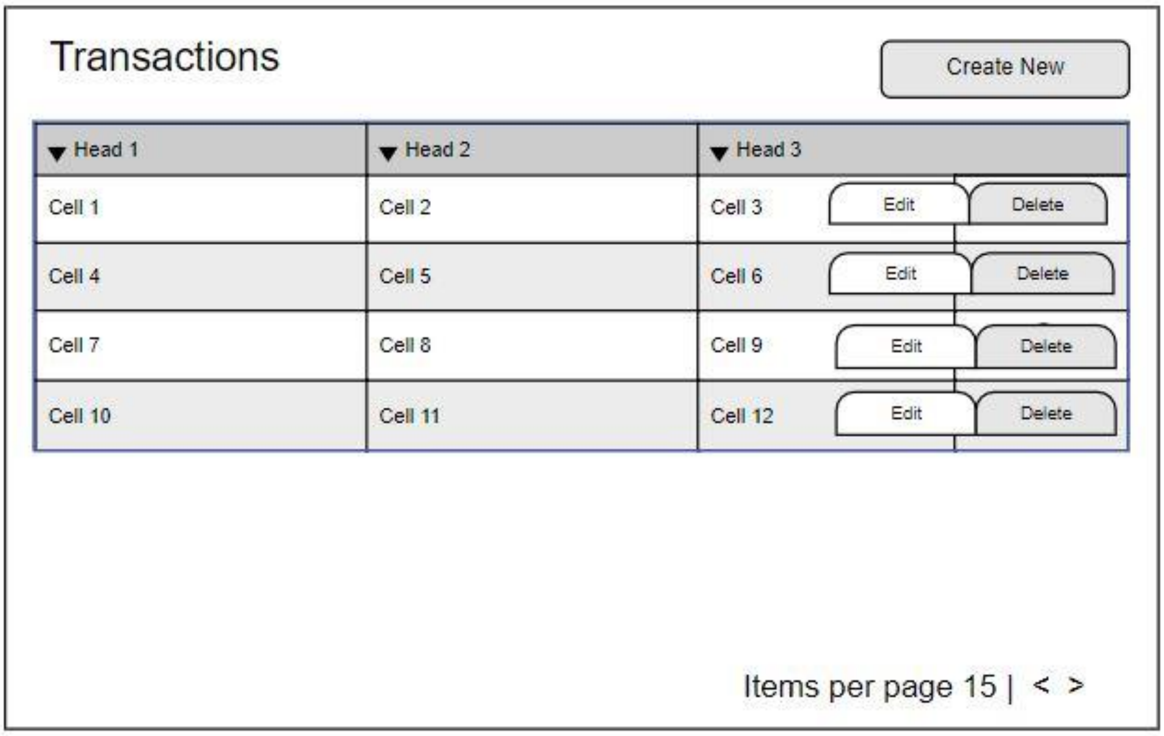
Amount

Description

Donation ▼

Save Transaction

This window will be used to update as well



▼ Head 1	▼ Head 2	▼ Head 3
Cell 1	Cell 2	Cell 3 Edit Delete
Cell 4	Cell 5	Cell 6 Edit Delete
Cell 7	Cell 8	Cell 9 Edit Delete
Cell 10	Cell 11	Cell 12 Edit Delete

View all transactions

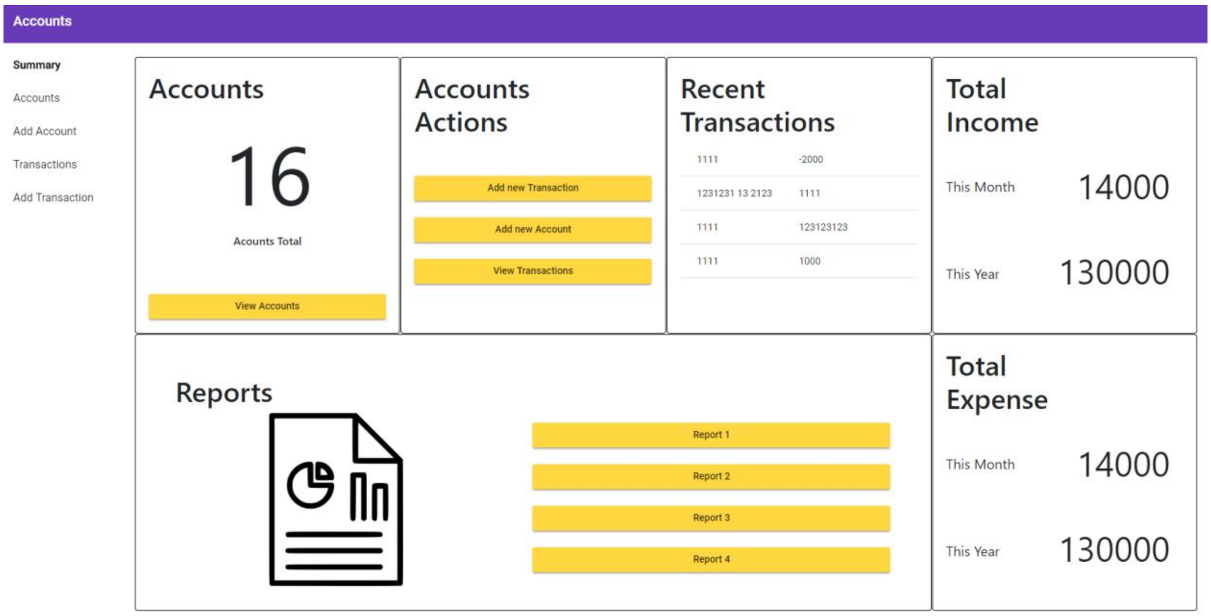
Commit ID: 7fdf234 account design

3.2 Task 2 Detailed Account Page Implementation

Estimate Time:	4 Hours
Actual Time:	5 Hours
Actual Time (this sprint):	5 Hours

Description

From this task I designed the GUIs of needed pages. I also completed incomplete GUIs that was unfinished in previous sprint (dashboard and view accounts GUIs).


Accounts

Summary
Accounts
Add Account
Transactions
Add Transaction

Accounts

16

Acounts Total

View Accounts

Accounts Actions

Add new Transaction
Add new Account
View Transactions

Recent Transactions

1111	-2000
1231231 13 2123	1111
1111	123123123
1111	1000

Total Income

This Month 14000

This Year 130000

Reports

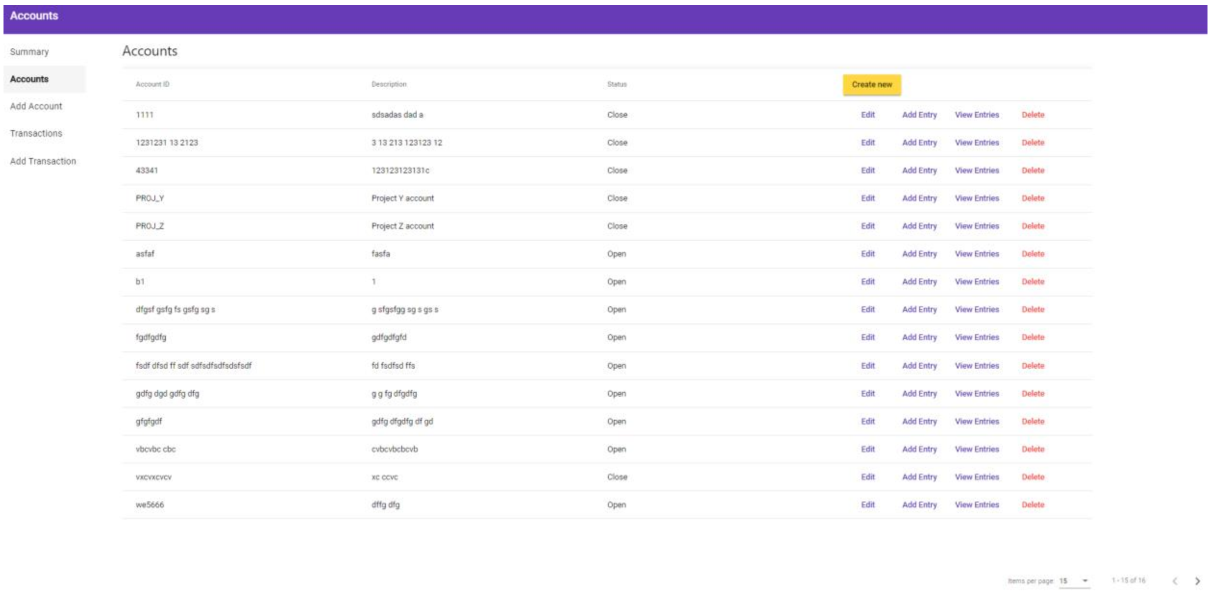
Report 1
Report 2
Report 3
Report 4

Total Expense

This Month 14000

This Year 130000

Dashboard


Accounts

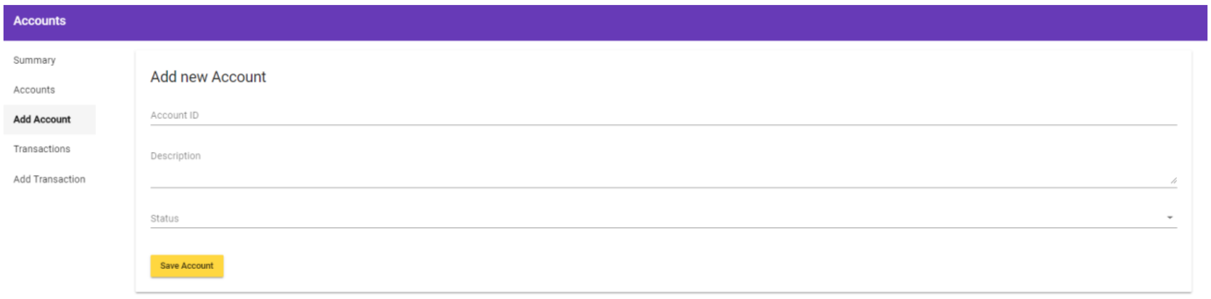
Summary
Accounts
Add Account
Transactions
Add Transaction

Accounts

Account ID	Description	Status	Create new			
1111	sdsadas dad a	Close	Edit	Add Entry	View Entries	Delete
1231231 13 2123	3 13 213 123123 12	Close	Edit	Add Entry	View Entries	Delete
43341	123123123131c	Close	Edit	Add Entry	View Entries	Delete
PROJ_Y	Project Y account	Close	Edit	Add Entry	View Entries	Delete
PROJ_Z	Project Z account	Close	Edit	Add Entry	View Entries	Delete
asfaf	fasfa	Open	Edit	Add Entry	View Entries	Delete
b1	1	Open	Edit	Add Entry	View Entries	Delete
dfgsf gsfg fs gsfg sg s	g sfgsfgg sg s gs s	Open	Edit	Add Entry	View Entries	Delete
fgdfgdfg	gdfgdfgfd	Open	Edit	Add Entry	View Entries	Delete
fsdf dfsd ff sdf sdfsdfsdfsdsfsdf	fd fsdfsd ffs	Open	Edit	Add Entry	View Entries	Delete
gdfg dgd gdfg dfg	g g fg dfgdfg	Open	Edit	Add Entry	View Entries	Delete
gfgfgdf	gdfg dfgdfg df gd	Open	Edit	Add Entry	View Entries	Delete
vbcvbc cbc	cvbcvbcbcvb	Open	Edit	Add Entry	View Entries	Delete
vxcvxcvcv	xc ccvc	Close	Edit	Add Entry	View Entries	Delete
we5666	dffg dfg	Open	Edit	Add Entry	View Entries	Delete

Items per page: 15   1 - 15 of 16

Accounts page


Accounts

Summary
Accounts
Add Account
Transactions
Add Transaction

Add new Account

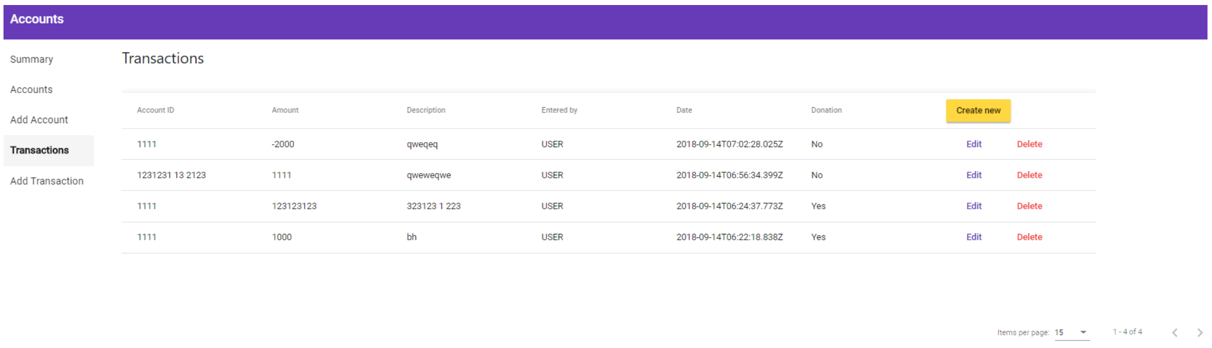
Account ID

Description

Status

Save Account

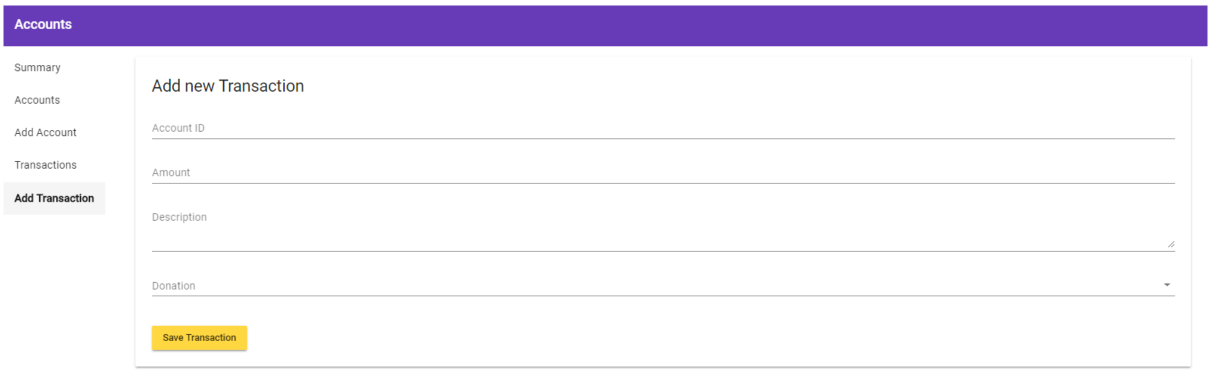
Add new Account page



Transactions

Account ID	Amount	Description	Entered by	Date	Donation		
1111	-2000	qweqeq	USER	2018-09-14T07:02:28.025Z	No	Edit	Delete
1231231 13 2123	1111	qweweqwe	USER	2018-09-14T06:56:34.399Z	No	Edit	Delete
1111	123123123	323123 1 223	USER	2018-09-14T06:24:37.773Z	Yes	Edit	Delete
1111	1000	bh	USER	2018-09-14T06:22:18.838Z	Yes	Edit	Delete

Transactions Page



Add new Transaction

Account ID	
Amount	
Description	
Donation	

Add new Transaction page

Commit IDs:

- f38bd25
- 9878303
- 2bb2467

3.3 Task 3 Detailed Account Page - Database Design

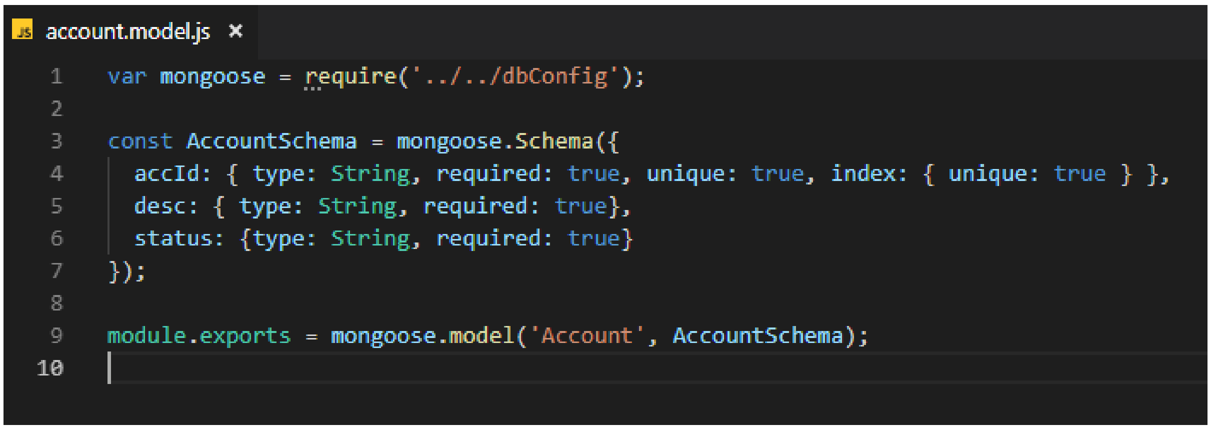
Estimate Time:	1 Hours
Actual Time:	1 Hours
Actual Time (this sprint):	1 Hours

Description

By this task I designed the database that need to store accounts and transactions, this involved finding needed attributes to correctly store relevant data.

For Account

- Account ID
- Description
- Status (Open, Close)

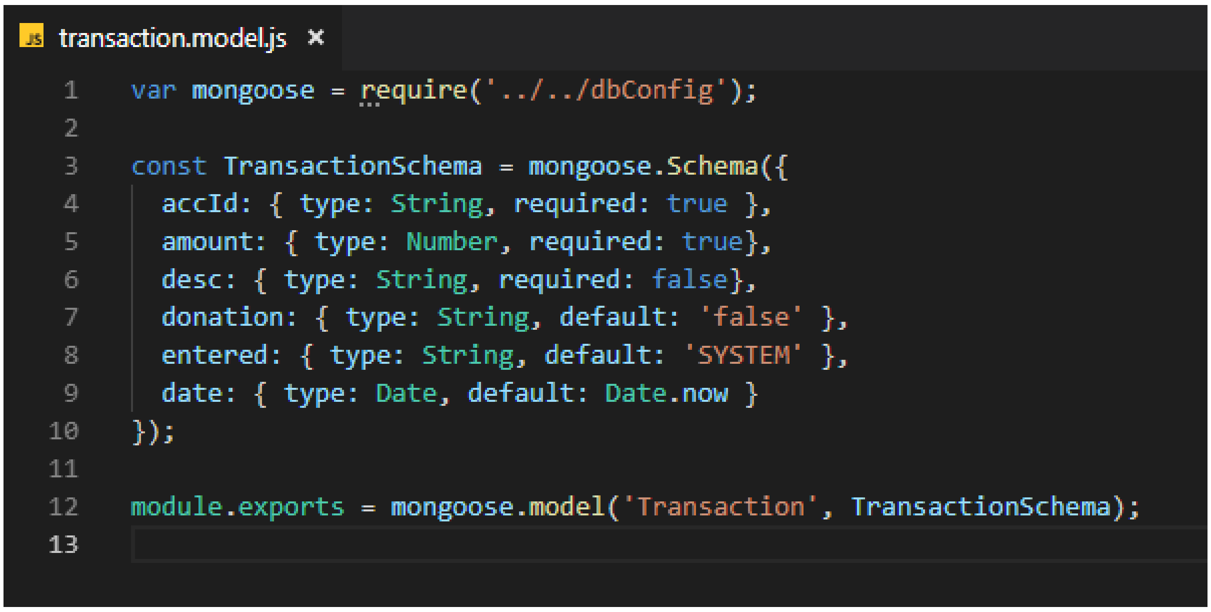


```
account.model.js ×
var mongoose = require('../../dbConfig');

const AccountSchema = mongoose.Schema({
  accId: { type: String, required: true, unique: true, index: { unique: true } },
  desc: { type: String, required: true},
  status: {type: String, required: true}
});

module.exports = mongoose.model('Account', AccountSchema);
```

Model created for Accounts in mongoDB

For Transaction

- ID
- Amount
- Description
- Donation (Yes, No)
- Entered By
- Entered date



```
transaction.model.js ×
var mongoose = require('../../dbConfig');

const TransactionSchema = mongoose.Schema({
  accId: { type: String, required: true },
  amount: { type: Number, required: true},
  desc: { type: String, required: false},
  donation: { type: String, default: 'false' },
  entered: { type: String, default: 'SYSTEM' },
  date: { type: Date, default: Date.now }
});

module.exports = mongoose.model('Transaction', TransactionSchema);
```

Model created for Transactions in mongoDB

Commit IDs:

- 9a010c7

3.4 Task 4 Detailed Account Page - Insert Method Calls

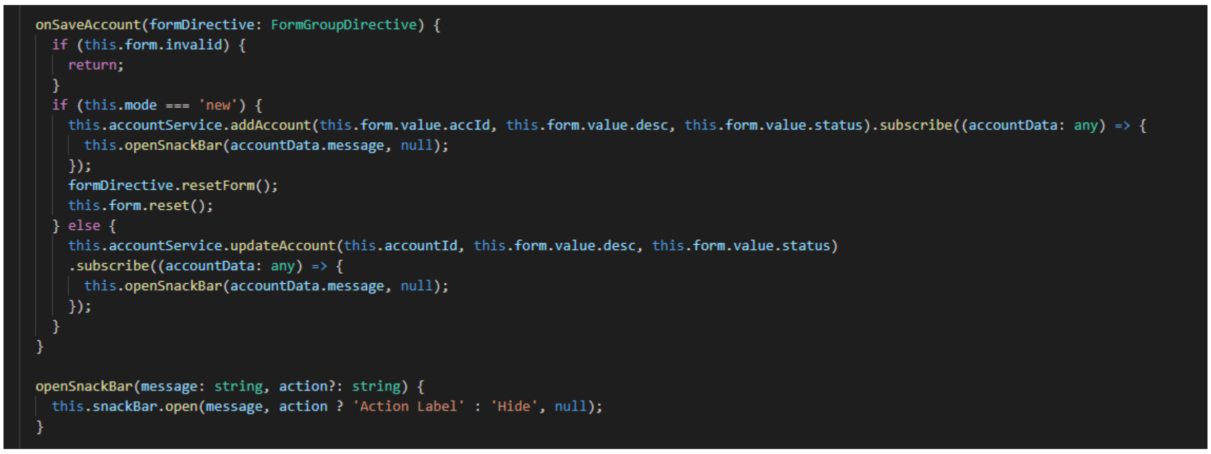
Estimate Time: 2 Hours

Actual Time: 2 Hours

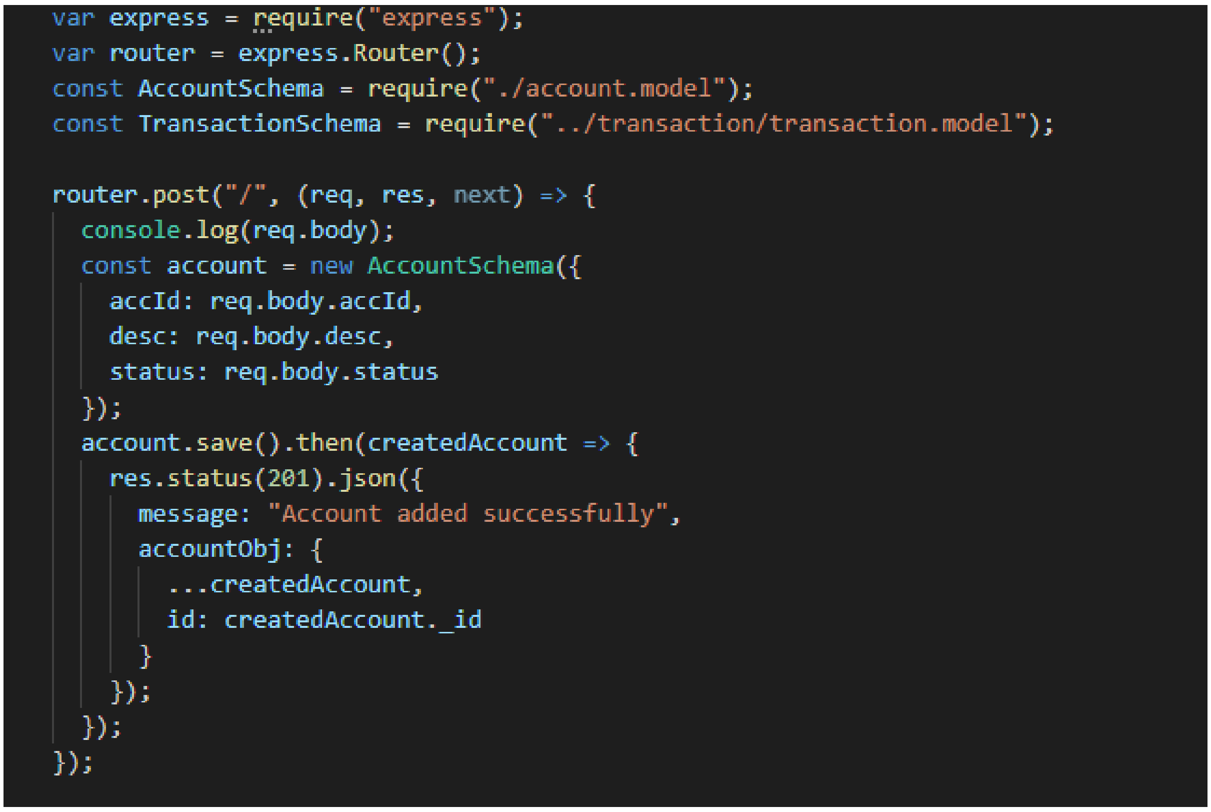
Actual Time (this sprint): 2 Hours

Description

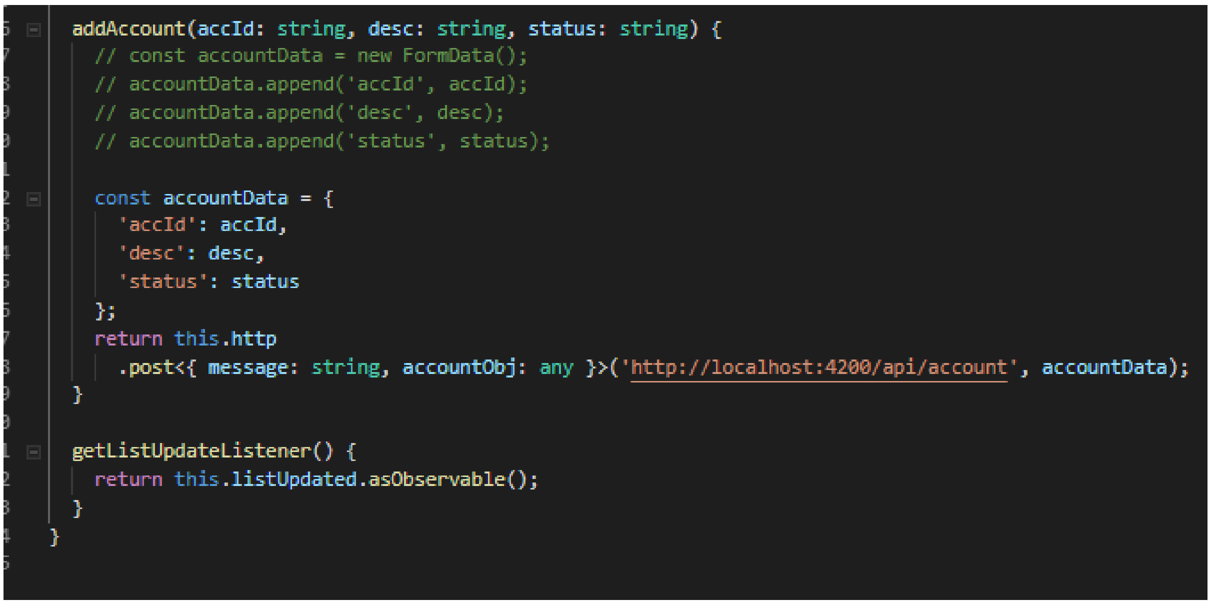
From this task I have Implemented the front end and middle tier code to insert data.



```
onSaveAccount(formDirective: FormGroupDirective) {
  if (this.form.invalid) {
    return;
  }
  if (this.mode === 'new') {
    this.accountService.addAccount(this.form.value.accId, this.form.value.desc, this.form.value.status).subscribe((accountData: any) => {
      this.openSnackBar(accountData.message, null);
    });
    formDirective.resetForm();
    this.form.reset();
  } else {
    this.accountService.updateAccount(this.accountId, this.form.value.desc, this.form.value.status)
    .subscribe((accountData: any) => {
      this.openSnackBar(accountData.message, null);
    });
  }
}

openSnackBar(message: string, action?: string) {
  this.snackBar.open(message, action ? 'Action Label' : 'Hide', null);
}
```



```
var express = require("express");
var router = express.Router();
const AccountSchema = require("./account.model");
const TransactionSchema = require("../transaction/transaction.model");

router.post("/", (req, res, next) => {
  console.log(req.body);
  const account = new AccountSchema({
    accId: req.body.accId,
    desc: req.body.desc,
    status: req.body.status
  });
  account.save().then(createdAccount => {
    res.status(201).json({
      message: "Account added successfully",
      accountObj: {
        ...createdAccount,
        id: createdAccount._id
      }
    });
  });
});
```



```
  addAccount(accId: string, desc: string, status: string) {
    // const accountData = new FormData();
    // accountData.append('accId', accId);
    // accountData.append('desc', desc);
    // accountData.append('status', status);

    const accountData = {
      'accId': accId,
      'desc': desc,
      'status': status
    };
    return this.http
      .post<{ message: string, accountObj: any }>('http://localhost:4200/api/account', accountData);
  }

  getListUpdateListener() {
    return this.listUpdated.asObservable();
  }
}
```

Commit IDs

- 698be7c

- Testing

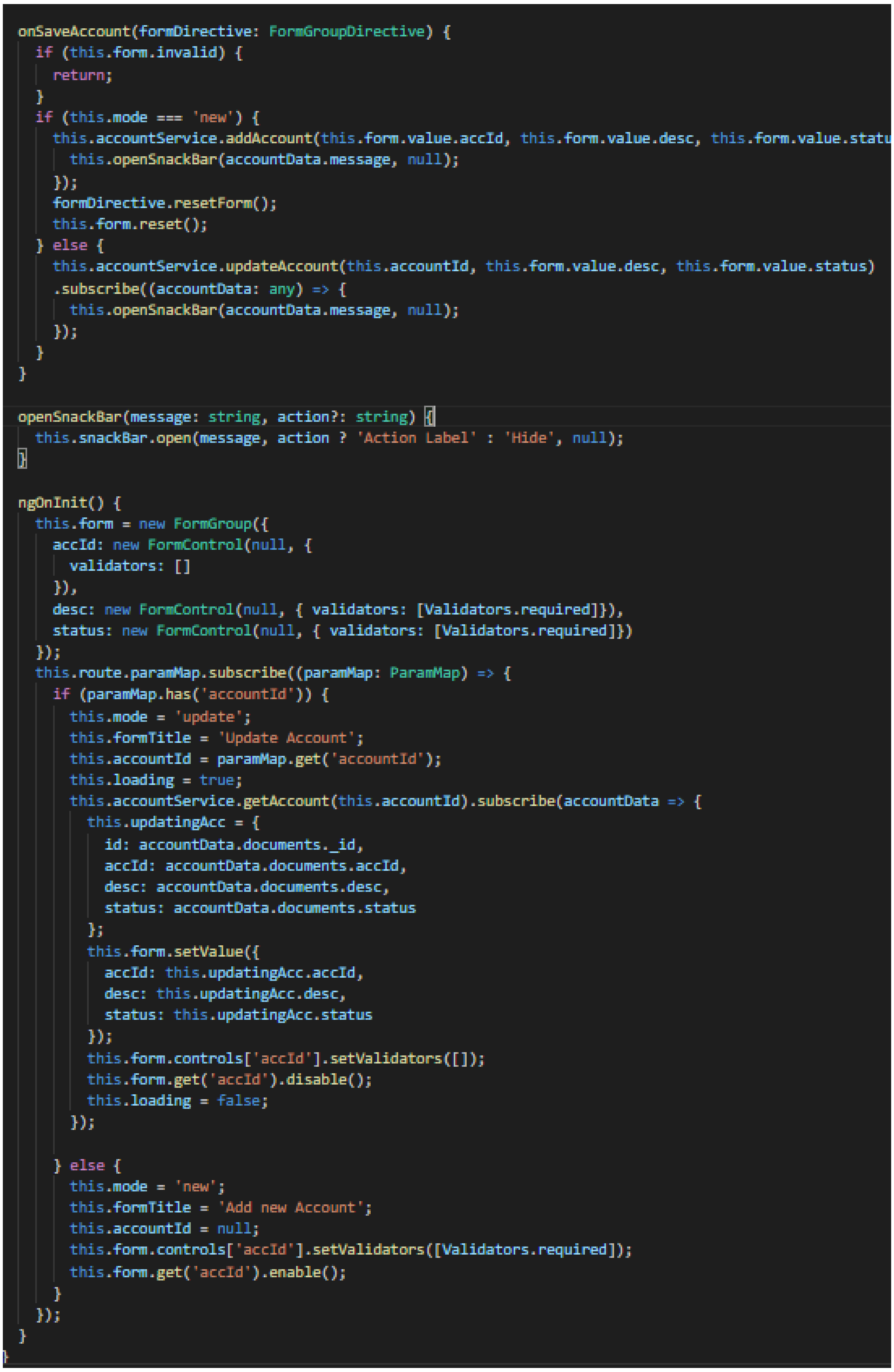
Test Case	Insert
Expected Behavior	Display success message
Test Steps	Goto add account page Insert data Press save
Test Status	Passed

3.5 Task 5 Detailed Account Page Update Method Calls

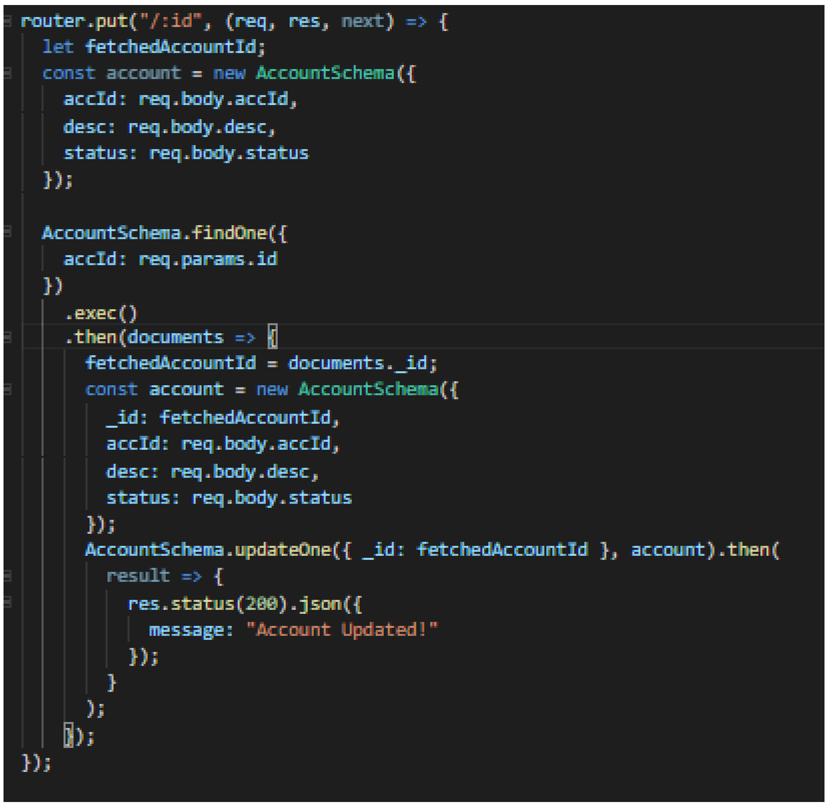
Estimate Time:	2 Hours
Actual Time:	2 Hours
Actual Time (this sprint):	2 Hours

Description

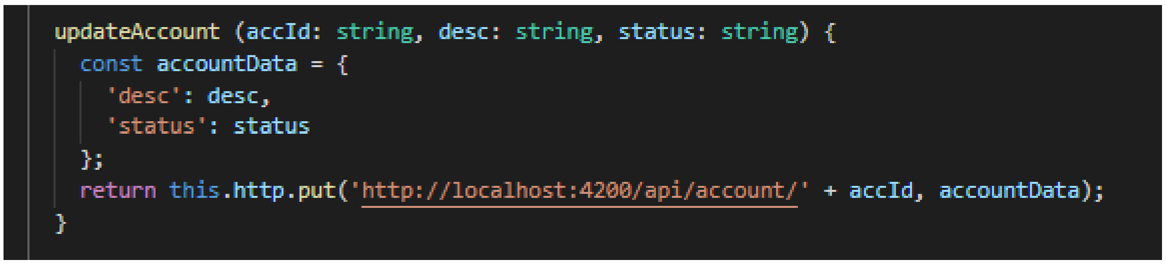
From this task I have Implemented the front end and middle tier code to update data.



```
  onSaveAccount(formDirective: FormGroupDirective) {
    if (this.form.invalid) {
      return;
    }
    if (this.mode === 'new') {
      this.accountService.addAccount(this.form.value.accId, this.form.value.desc, this.form.value.statu
        this.openSnackBar(accountData.message, null);
      });
      formDirective.resetForm();
      this.form.reset();
    } else {
      this.accountService.updateAccount(this.accountId, this.form.value.desc, this.form.value.status)
      .subscribe((accountData: any) => {
        this.openSnackBar(accountData.message, null);
      });
    }
  }

  openSnackBar(message: string, action?: string) {
    this.snackBar.open(message, action ? 'Action Label' : 'Hide', null);
  }

  ngOnInit() {
    this.form = new FormGroup({
      accId: new FormControl(null, {
        validators: []
      }),
      desc: new FormControl(null, { validators: [Validators.required]}),
      status: new FormControl(null, { validators: [Validators.required]})
    });
    this.route.paramMap.subscribe((paramMap: ParamMap) => {
      if (paramMap.has('accountId')) {
        this.mode = 'update';
        this.formTitle = 'Update Account';
        this.accountId = paramMap.get('accountId');
        this.loading = true;
        this.accountService.getAccount(this.accountId).subscribe(accountData => {
          this.updatingAcc = {
            id: accountData.documents._id,
            accId: accountData.documents.accId,
            desc: accountData.documents.desc,
            status: accountData.documents.status
          };
          this.form.setValue({
            accId: this.updatingAcc.accId,
            desc: this.updatingAcc.desc,
            status: this.updatingAcc.status
          });
          this.form.controls['accId'].setValidators([]);
          this.form.get('accId').disable();
          this.loading = false;
        });

      } else {
        this.mode = 'new';
        this.formTitle = 'Add new Account';
        this.accountId = null;
        this.form.controls['accId'].setValidators([Validators.required]);
        this.form.get('accId').enable();
      }
    });
  }
}
```

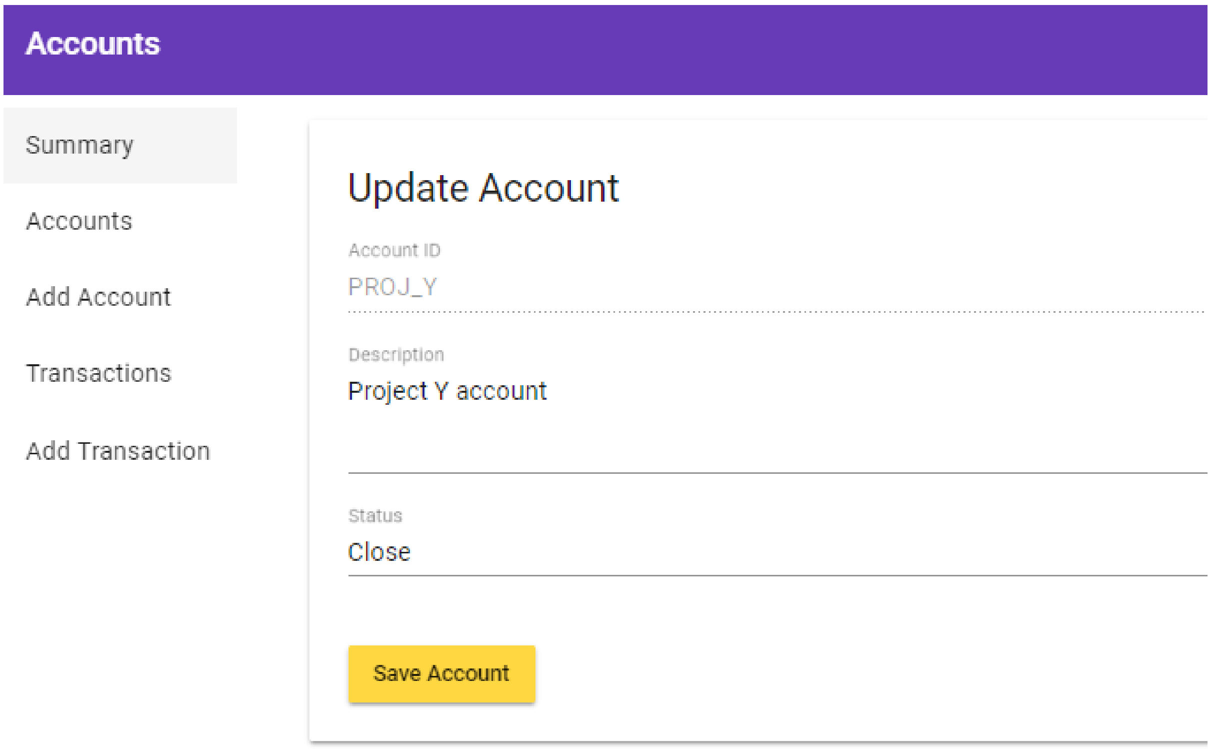


```
router.put("/:id", (req, res, next) => {
  let fetchedAccountId;
  const account = new AccountSchema({
    accId: req.body.accId,
    desc: req.body.desc,
    status: req.body.status
  });

  AccountSchema.findOne({
    accId: req.params.id
  })
    .exec()
    .then(documents => {
      fetchedAccountId = documents._id;
      const account = new AccountSchema({
        _id: fetchedAccountId,
        accId: req.body.accId,
        desc: req.body.desc,
        status: req.body.status
      });
      AccountSchema.updateOne({ _id: fetchedAccountId }, account).then(
        result => {
          res.status(200).json({
            message: "Account Updated!"
          });
        }
      );
    });
});
```



```
  updateAccount (accId: string, desc: string, status: string) {
    const accountData = {
      'desc': desc,
      'status': status
    };
    return this.http.put('http://localhost:4200/api/account/' + accId, accountData);
  }
```

I Also dynamically changed the add account page to update.



Update Account

Account ID
PROJ_Y

Description
Project Y account

Status
Close

Commit IDs:

- 42ac9fc

- Testing

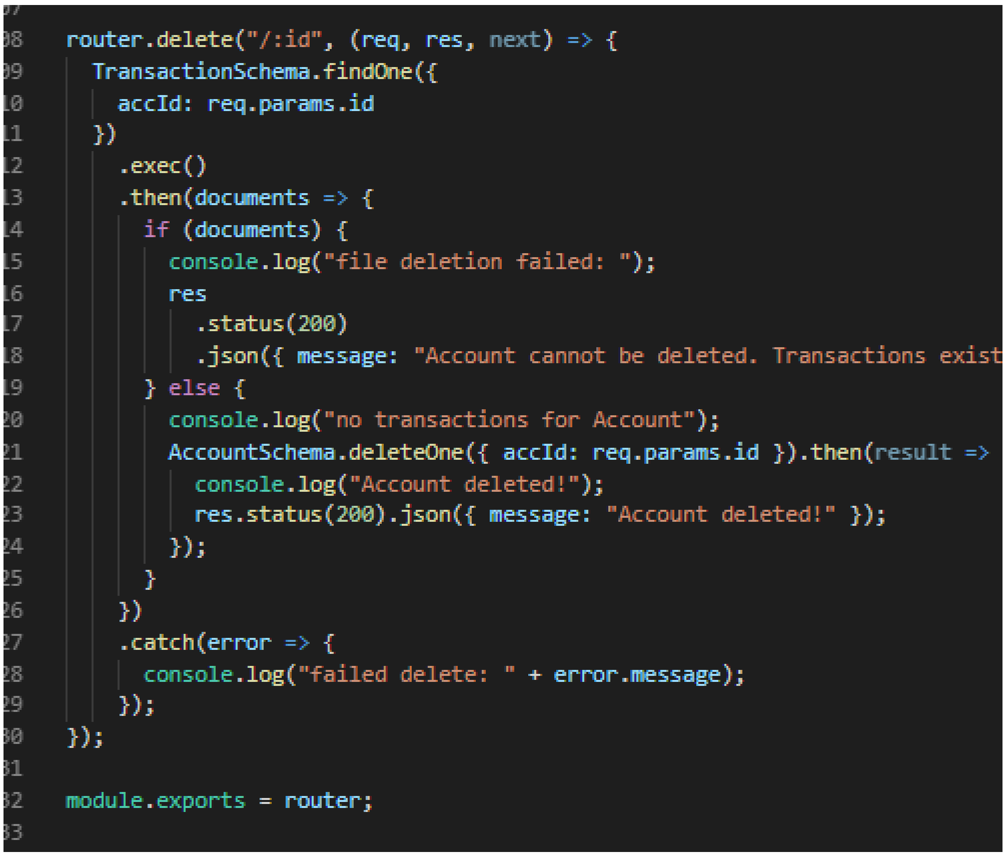
Test Case	Update
Expected Behavior	Display Update Success message
Test Steps	Select edit button on accounts page Edit data Press save
Test Status	Passed

3.6 Task 6 Detailed Account Page Delete Method Calls

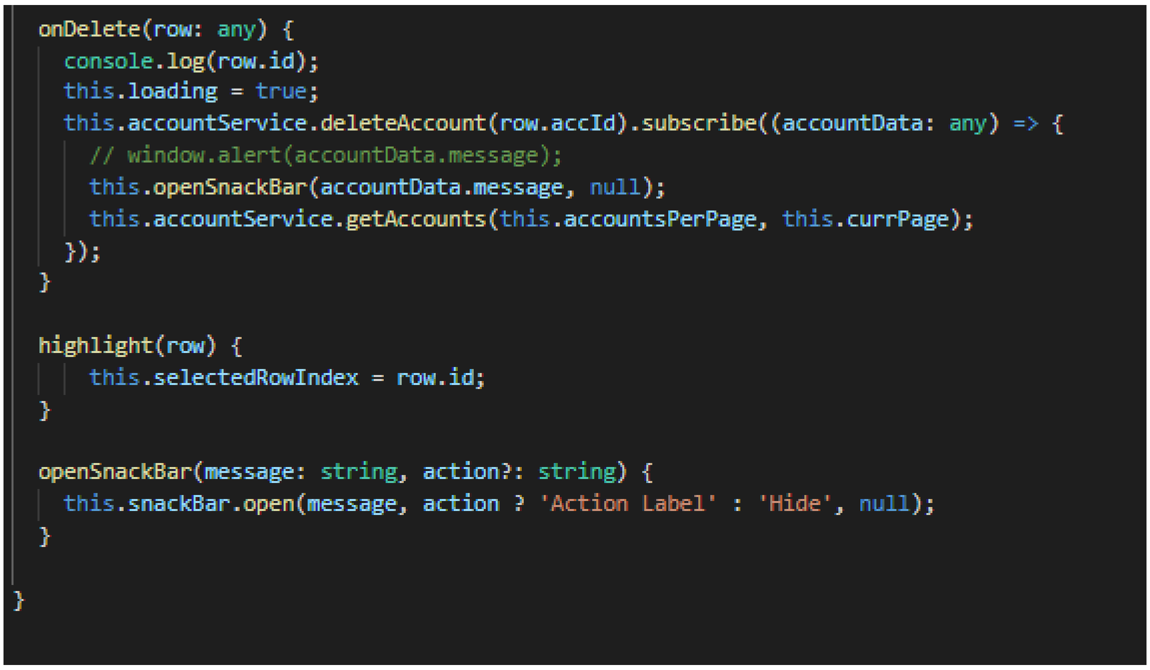
Estimate Time:	2 Hours
Actual Time:	2 Hours
Actual Time (this sprint):	2 Hours

Description

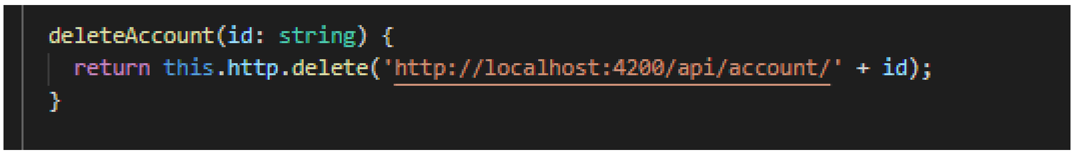
From this task I have Implemented the front end and middle tier code to delete data.



```
router.delete("/:id", (req, res, next) => {
  TransactionSchema.findOne({
    accId: req.params.id
  })
    .exec()
    .then(documents => {
      if (documents) {
        console.log("file deletion failed: ");
        res
          .status(200)
          .json({ message: "Account cannot be deleted. Transactions exist
      } else {
        console.log("no transactions for Account");
        AccountSchema.deleteOne({ accId: req.params.id }).then(result =>
          console.log("Account deleted!");
          res.status(200).json({ message: "Account deleted!" });
        });
      }
    })
    .catch(error => {
      console.log("failed delete: " + error.message);
    });
});

module.exports = router;
```



```
  onDelete(row: any) {
    console.log(row.id);
    this.loading = true;
    this.accountService.deleteAccount(row.accId).subscribe((accountData: any) => {
      // window.alert(accountData.message);
      this.openSnackBar(accountData.message, null);
      this.accountService.getAccounts(this.accountsPerPage, this.currPage);
    });
  }

  highlight(row) {
      this.selectedRowIndex = row.id;
  }

  openSnackBar(message: string, action?: string) {
    this.snackBar.open(message, action ? 'Action Label' : 'Hide', null);
  }

}
```



```
  deleteAccount(id: string) {
    return this.http.delete('http://localhost:4200/api/account/' + id);
  }
```

- Testing

Test Case	Delete

Expected Behavior	Display delete message Record disappear
Test Steps	Press delete on Accounts page
Test Status	Passed

3.7 Task 7 Detailed Account Page Validation Methods

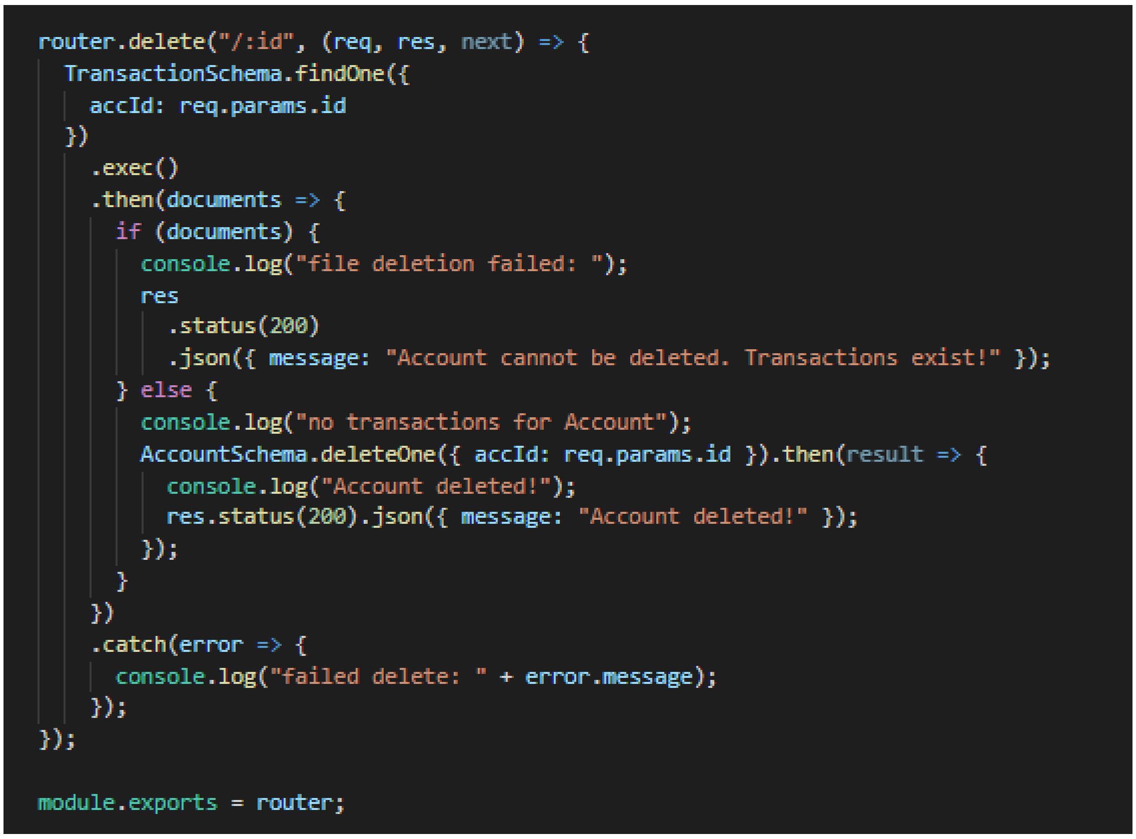
Estimate Time: 2 Hours

Actual Time: 2 Hours

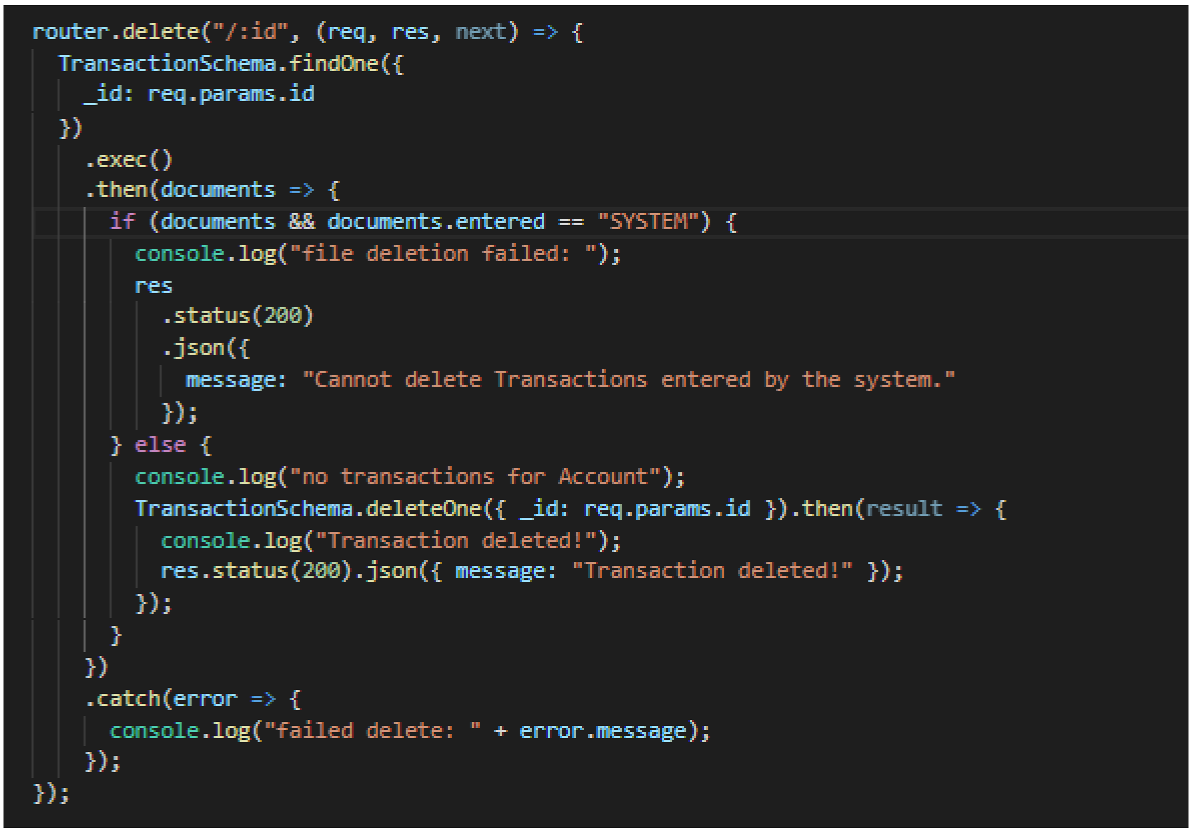
Actual Time (this sprint): 2 Hours

Description

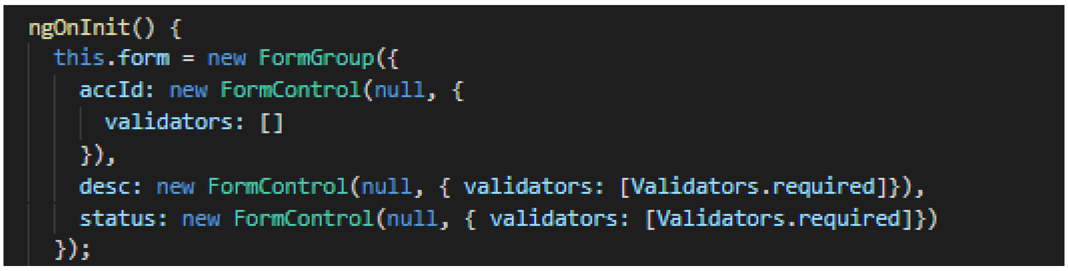
From this task I added front end and middle tier validations to accounts and transactions.



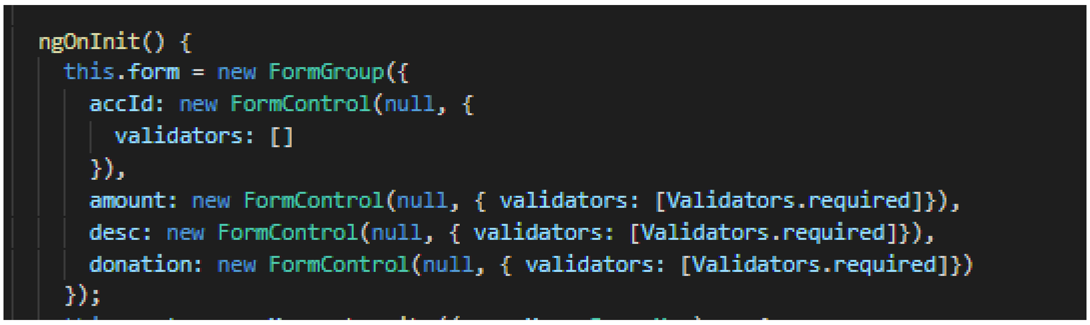
```
router.delete("/:id", (req, res, next) => {
  TransactionSchema.findOne({
    accId: req.params.id
  })
    .exec()
    .then(documents => {
      if (documents) {
        console.log("file deletion failed: ");
        res
          .status(200)
          .json({ message: "Account cannot be deleted. Transactions exist!" });
      } else {
        console.log("no transactions for Account");
        AccountSchema.deleteOne({ accId: req.params.id }).then(result => {
          console.log("Account deleted!");
          res.status(200).json({ message: "Account deleted!" });
        });
      }
    })
    .catch(error => {
      console.log("failed delete: " + error.message);
    });
});

module.exports = router;
```



```
router.delete("/:id", (req, res, next) => {
  TransactionSchema.findOne({
    _id: req.params.id
  })
    .exec()
    .then(documents => {
      if (documents && documents.entered == "SYSTEM") {
        console.log("file deletion failed: ");
        res
          .status(200)
          .json({
            message: "Cannot delete Transactions entered by the system."
          });
      } else {
        console.log("no transactions for Account");
        TransactionSchema.deleteOne({ _id: req.params.id }).then(result => {
          console.log("Transaction deleted!");
          res.status(200).json({ message: "Transaction deleted!" });
        });
      }
    })
    .catch(error => {
      console.log("failed delete: " + error.message);
    });
});
```



```
ngOnInit() {
  this.form = new FormGroup({
    accId: new FormControl(null, {
      validators: []
    }),
    desc: new FormControl(null, { validators: [Validators.required]}),
    status: new FormControl(null, { validators: [Validators.required]})
  });
```



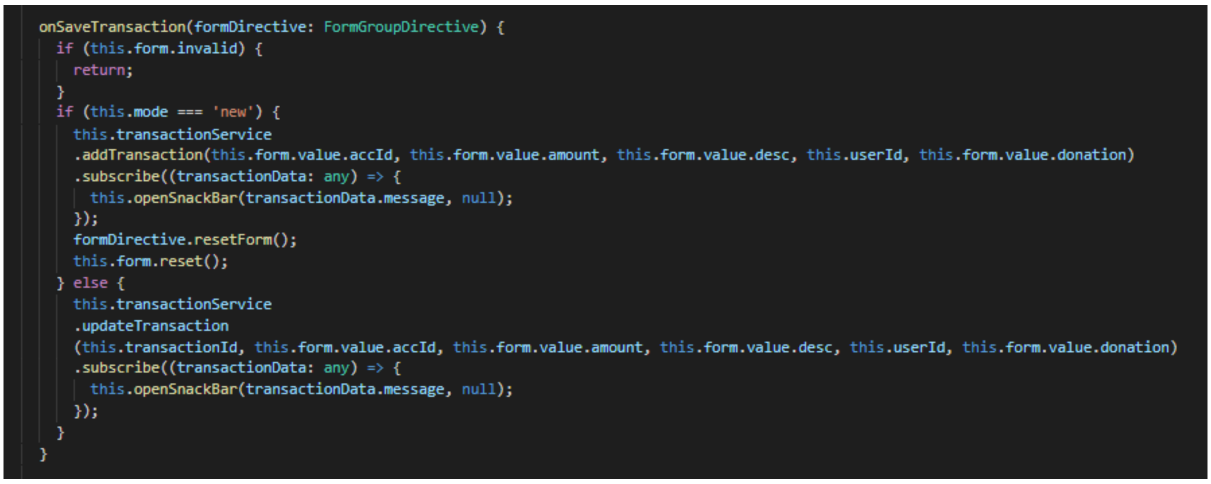
```
ngOnInit() {
  this.form = new FormGroup({
    accId: new FormControl(null, {
      validators: []
    }),
    amount: new FormControl(null, { validators: [Validators.required]}),
    desc: new FormControl(null, { validators: [Validators.required]}),
    donation: new FormControl(null, { validators: [Validators.required]})
  });
```

3.8 Task 8 Detailed Account Page Transaction Table Insert Methods

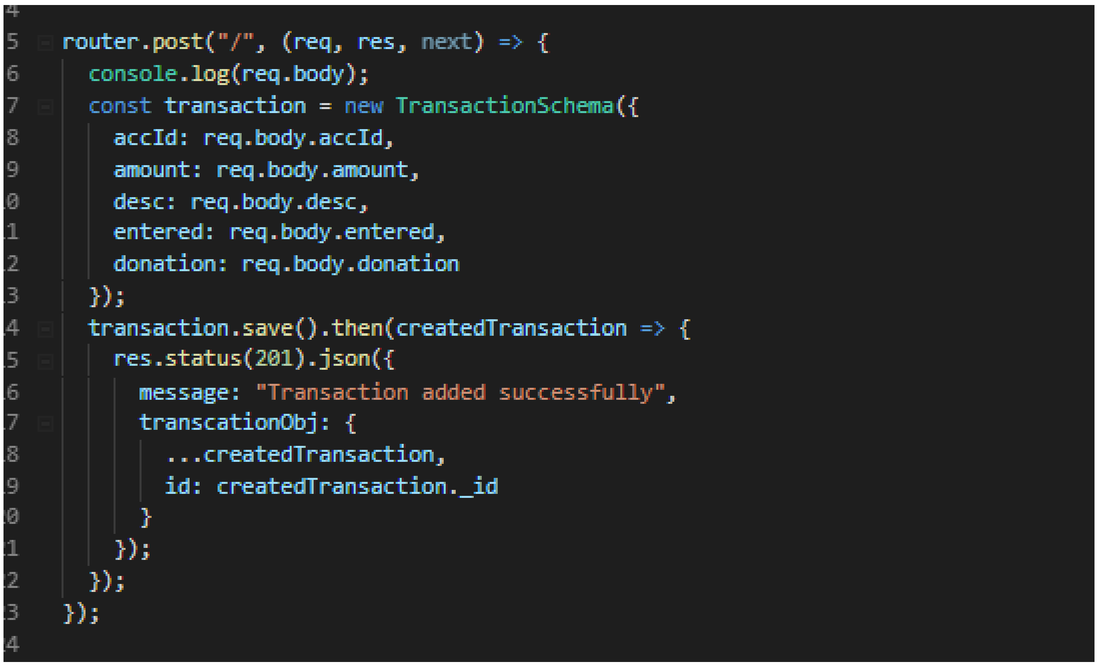
Estimate Time:	2 Hours
Actual Time:	2 Hours
Actual Time (this sprint):	2 Hours

Description

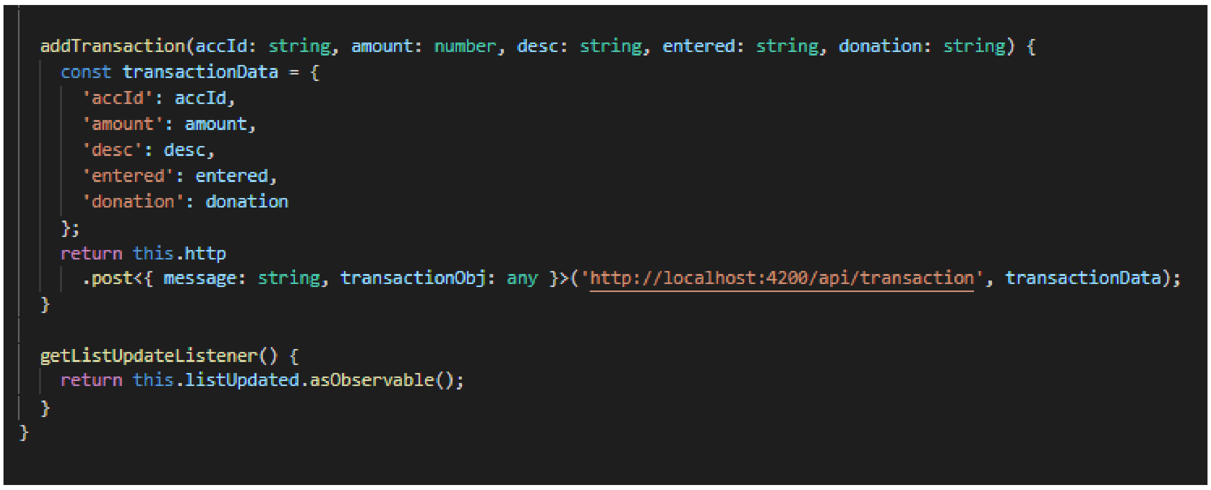
From this task I have Implemented the front end and middle tier code to insert data.



```
onSaveTransaction(formDirective: FormGroupDirective) {
  if (this.form.invalid) {
    return;
  }
  if (this.mode === 'new') {
    this.transactionService
    .addTransaction(this.form.value.accId, this.form.value.amount, this.form.value.desc, this.userId, this.form.value.donation)
    .subscribe((transactionData: any) => {
      this.openSnackBar(transactionData.message, null);
    });
    formDirective.resetForm();
    this.form.reset();
  } else {
    this.transactionService
    .updateTransaction
    (this.transactionId, this.form.value.accId, this.form.value.amount, this.form.value.desc, this.userId, this.form.value.donation)
    .subscribe((transactionData: any) => {
      this.openSnackBar(transactionData.message, null);
    });
  }
}
```



```
router.post("/", (req, res, next) => {
  console.log(req.body);
  const transaction = new TransactionSchema({
    accId: req.body.accId,
    amount: req.body.amount,
    desc: req.body.desc,
    entered: req.body.entered,
    donation: req.body.donation
  });
  transaction.save().then(createdTransaction => {
    res.status(201).json({
      message: "Transaction added successfully",
      transcationObj: {
        ...createdTransaction,
        id: createdTransaction._id
      }
    });
  });
});
```



```
  addTransaction(accId: string, amount: number, desc: string, entered: string, donation: string) {
    const transactionData = {
      'accId': accId,
      'amount': amount,
      'desc': desc,
      'entered': entered,
      'donation': donation
    };
    return this.http
      .post<{ message: string, transactionObj: any }>('http://localhost:4200/api/transaction', transactionData);
  }

  getListUpdateListener() {
    return this.listUpdated.asObservable();
  }
}
```

- Testing

Test Case	Insert

Expected Behavior	Display success message
Test Steps	Goto add transactions page Insert data Press save
Test Status	Passed

3.9 Task 9 Detailed Account Page Transaction Table Update Methods

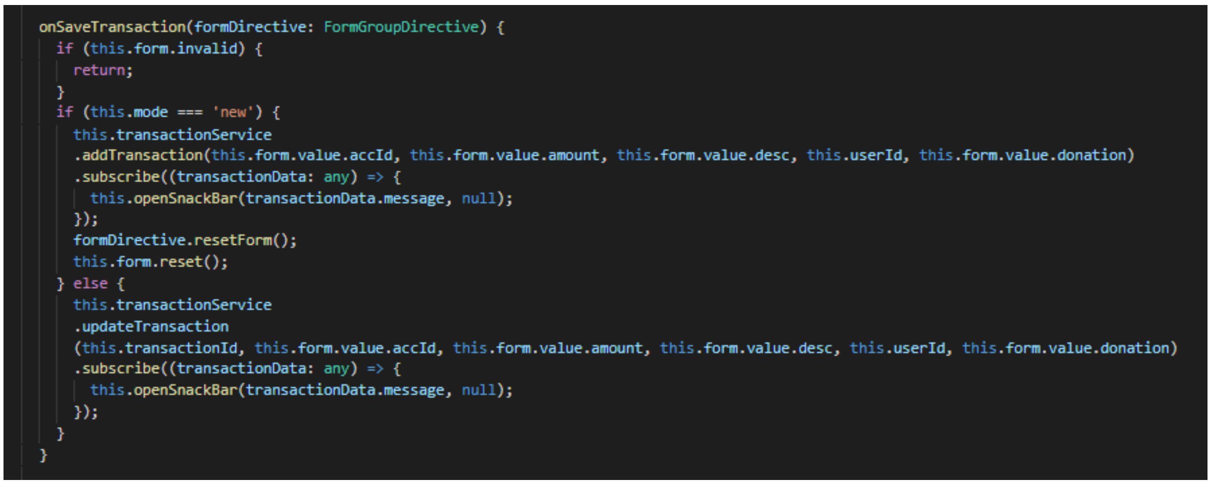
Estimate Time: 2 Hours

Actual Time: 2 Hours

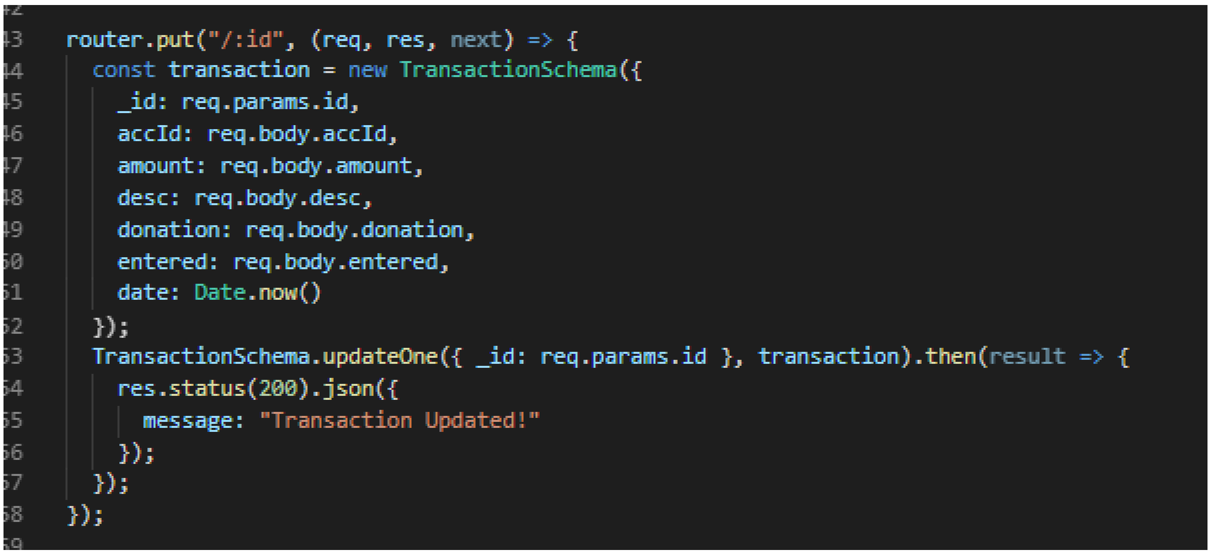
Actual Time (this sprint): 2 Hours

Description

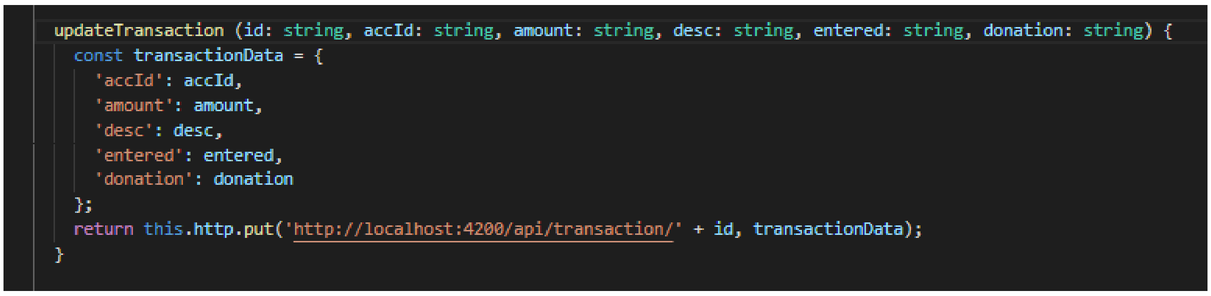
From this task I have Implemented the front end and middle tier code to update data.



```
onSaveTransaction(formDirective: FormGroupDirective) {
  if (this.form.invalid) {
    return;
  }
  if (this.mode === 'new') {
    this.transactionService
    .addTransaction(this.form.value.accId, this.form.value.amount, this.form.value.desc, this.userId, this.form.value.donation)
    .subscribe((transactionData: any) => {
      this.openSnackBar(transactionData.message, null);
    });
    formDirective.resetForm();
    this.form.reset();
  } else {
    this.transactionService
    .updateTransaction
    (this.transactionId, this.form.value.accId, this.form.value.amount, this.form.value.desc, this.userId, this.form.value.donation)
    .subscribe((transactionData: any) => {
      this.openSnackBar(transactionData.message, null);
    });
  }
}
```

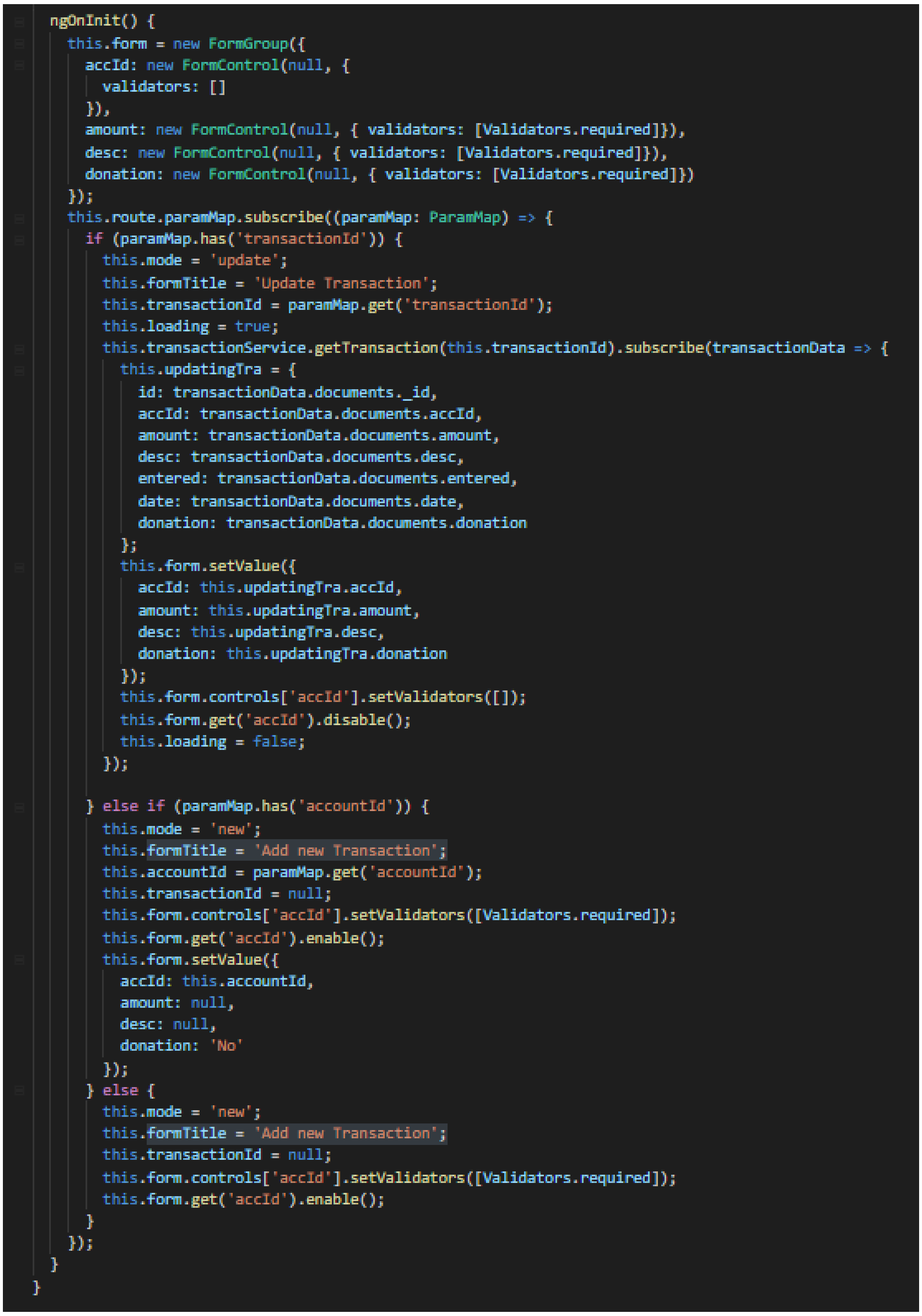


```
router.put("/:id", (req, res, next) => {
  const transaction = new TransactionSchema({
    _id: req.params.id,
    accId: req.body.accId,
    amount: req.body.amount,
    desc: req.body.desc,
    donation: req.body.donation,
    entered: req.body.entered,
    date: Date.now()
  });
  TransactionSchema.updateOne({ _id: req.params.id }, transaction).then(result => {
    res.status(200).json({
      message: "Transaction Updated!"
    });
  });
});
```

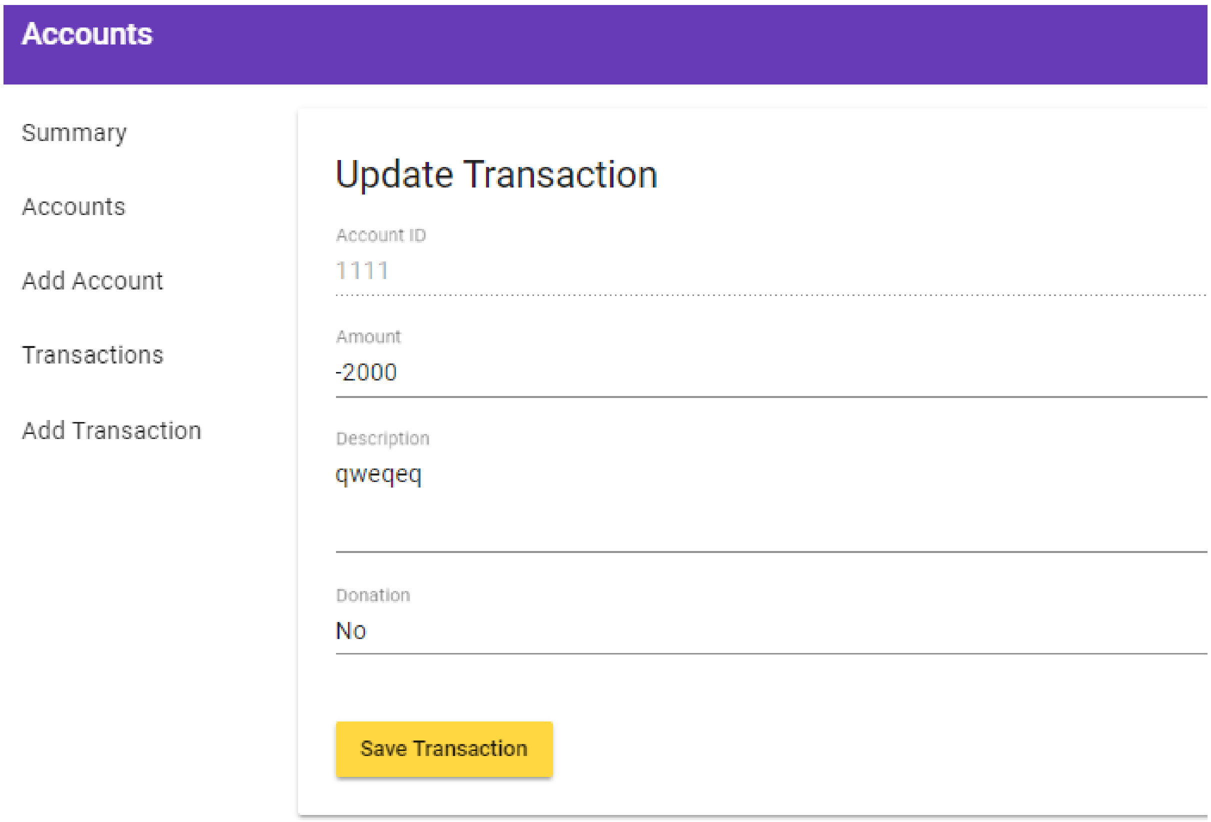


```
updateTransaction (id: string, accId: string, amount: string, desc: string, entered: string, donation: string) {
  const transactionData = {
    'accId': accId,
    'amount': amount,
    'desc': desc,
    'entered': entered,
    'donation': donation
  };
  return this.http.put('http://localhost:4200/api/transaction/' + id, transactionData);
}
```

I Also dynamically changed the add transaction page to update.



```
  ngOnInit() {
    this.form = new FormGroup({
      accId: new FormControl(null, {
        validators: []
      }),
      amount: new FormControl(null, { validators: [Validators.required]}),
      desc: new FormControl(null, { validators: [Validators.required]}),
      donation: new FormControl(null, { validators: [Validators.required]})
    });
    this.route.paramMap.subscribe((paramMap: ParamMap) => {
      if (paramMap.has('transactionId')) {
        this.mode = 'update';
        this.formTitle = 'Update Transaction';
        this.transactionId = paramMap.get('transactionId');
        this.loading = true;
        this.transactionService.getTransaction(this.transactionId).subscribe(transactionData => {
          this.updatingTra = {
            id: transactionData.documents._id,
            accId: transactionData.documents.accId,
            amount: transactionData.documents.amount,
            desc: transactionData.documents.desc,
            entered: transactionData.documents.entered,
            date: transactionData.documents.date,
            donation: transactionData.documents.donation
          };
          this.form.setValue({
            accId: this.updatingTra.accId,
            amount: this.updatingTra.amount,
            desc: this.updatingTra.desc,
            donation: this.updatingTra.donation
          });
          this.form.controls['accId'].setValidators([]);
          this.form.get('accId').disable();
          this.loading = false;
        });

      } else if (paramMap.has('accountId')) {
        this.mode = 'new';
        this.formTitle = 'Add new Transaction';
        this.accountId = paramMap.get('accountId');
        this.transactionId = null;
        this.form.controls['accId'].setValidators([Validators.required]);
        this.form.get('accId').enable();
        this.form.setValue({
          accId: this.accountId,
          amount: null,
          desc: null,
          donation: 'No'
        });
      } else {
        this.mode = 'new';
        this.formTitle = 'Add new Transaction';
        this.transactionId = null;
        this.form.controls['accId'].setValidators([Validators.required]);
        this.form.get('accId').enable();
      }
    });
  }
}
```



Update Transaction

Account ID
1111

Amount
-2000

Description
qweqeq

Donation
No

- Testing

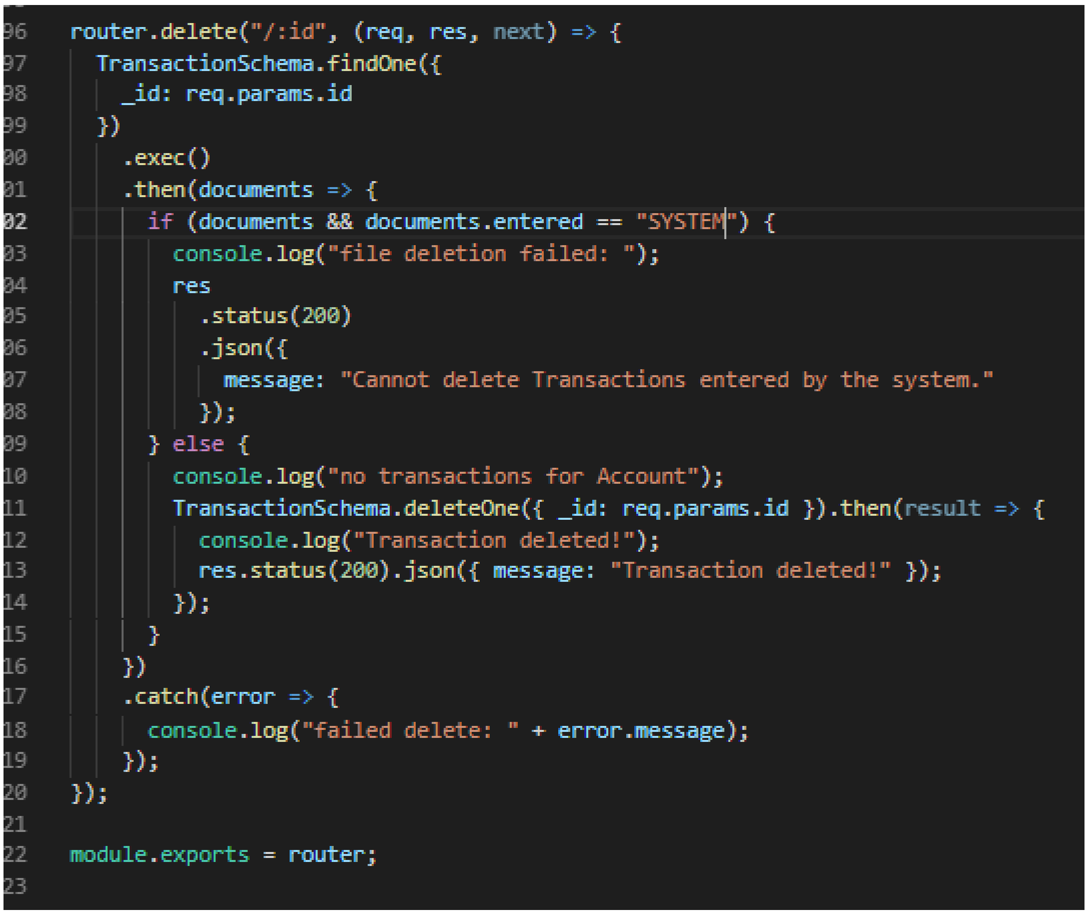
Test Case	Update
Expected Behavior	Display Update Success message
Test Steps	Select edit button on transaction page Edit data Press save
Test Status	Passed

3.10 Task 10 Detailed Account Page Transaction Table Delete Methods

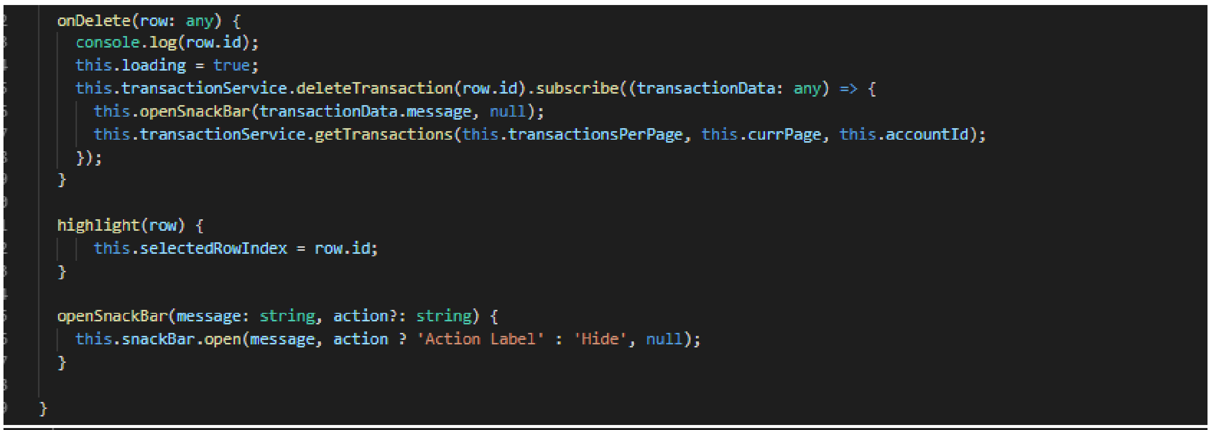
Estimate Time:	2 Hours
Actual Time:	2 Hours
Actual Time (this sprint):	2 Hours

Description

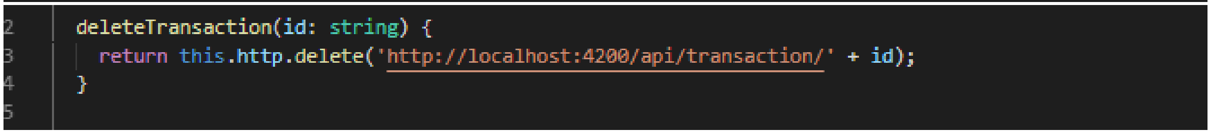
From this task I have Implemented the front end and middle tier code to delete data.



```
router.delete("/:id", (req, res, next) => {
  TransactionSchema.findOne({
    _id: req.params.id
  })
    .exec()
    .then(documents => {
      if (documents && documents.entered == "SYSTEM") {
        console.log("file deletion failed: ");
        res
          .status(200)
          .json({
            message: "Cannot delete Transactions entered by the system."
          });
      } else {
        console.log("no transactions for Account");
        TransactionSchema.deleteOne({ _id: req.params.id }).then(result => {
          console.log("Transaction deleted!");
          res.status(200).json({ message: "Transaction deleted!" });
        });
      }
    })
    .catch(error => {
      console.log("failed delete: " + error.message);
    });
});

module.exports = router;
```



```
  onDelete(row: any) {
    console.log(row.id);
    this.loading = true;
    this.transactionService.deleteTransaction(row.id).subscribe((transactionData: any) => {
      this.openSnackBar(transactionData.message, null);
      this.transactionService.getTransactions(this.transactionsPerPage, this.currPage, this.accountId);
    });
  }

  highlight(row) {
      this.selectedRowIndex = row.id;
  }

  openSnackBar(message: string, action?: string) {
    this.snackBar.open(message, action ? 'Action Label' : 'Hide', null);
  }

}
```



```
  deleteTransaction(id: string) {
    return this.http.delete('http://localhost:4200/api/transaction/' + id);
  }
```

- Testing

Test Case	Delete
Expected Behavior	Display delete message Record disappear
Test Steps	Press delete on transactions page
Test Status	Passed

3.11 Task 11 Sprint 4 Planning

Estimate Time:	1 Hours
Actual Time:	1 Hours
Actual Time (this sprint):	1 Hours

Description

This task is to plan upcoming sprint.

Please find the sprint 4 planning meeting minutes from the following link.

Bitbucket Link:

https://bitbucket.org/Computing_Projects_SLIIT/2018_sd07/src/master/Documents/Sprint%20Documents/Sprint%204/Sprint%204%20Planning%20Minutes.pdf

3.12 Task 12 Discuss design issues Sprint 3

Estimate Time:	1 Hours
Actual Time:	1 Hours
Actual Time (this sprint):	1 Hours

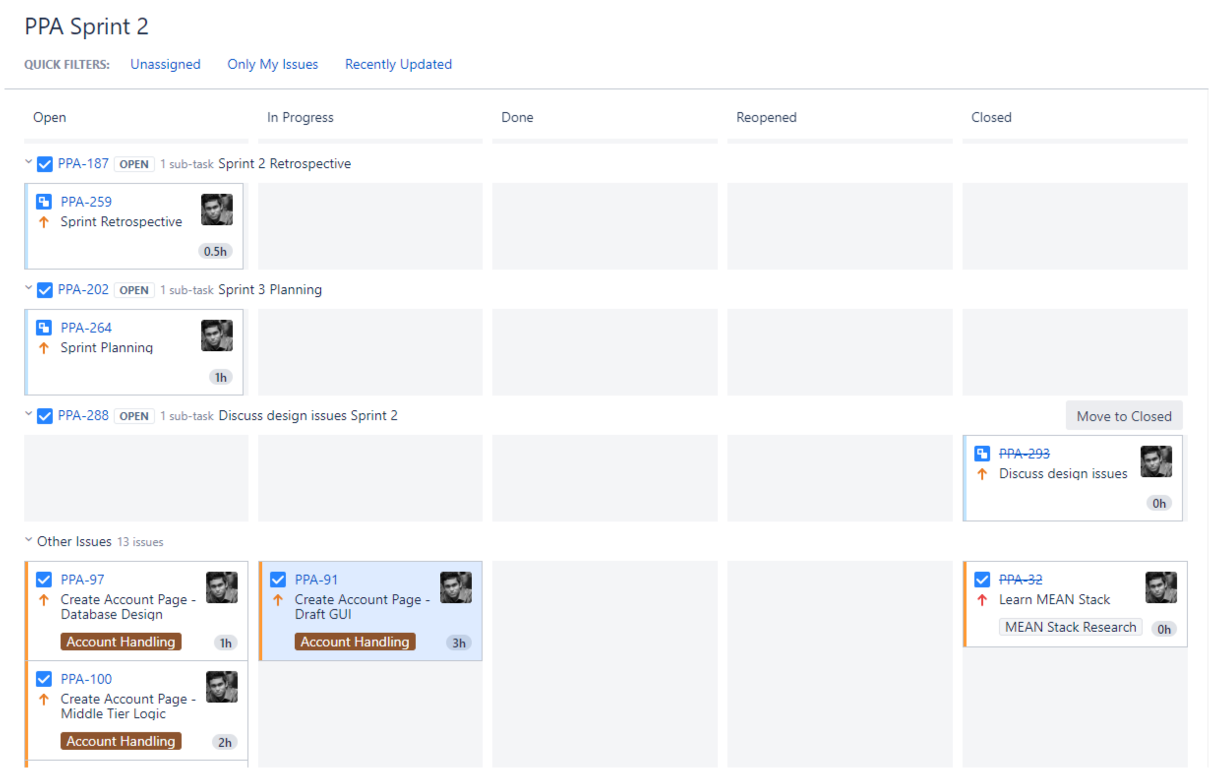
Description

This task is to discuss and solve technical and design issues.

4. Development Methodology

4.1 Minutes

We use Jira as a project management tool. Here is my issue dashboard.



PPA Sprint 2

QUICK FILTERS: Unassigned Only My Issues Recently Updated

Open	In Progress	Done	Reopened	Closed
PPA-187 OPEN 1 sub-task Sprint 2 Retrospective				
PPA-259 Sprint Retrospective 0.5h				
PPA-202 OPEN 1 sub-task Sprint 3 Planning				
PPA-264 Sprint Planning 1h				
PPA-288 OPEN 1 sub-task Discuss design issues Sprint 2				Move to Closed
				~~PPA-293~~ Discuss design issues 0h
Other Issues 13 issues				
PPA-97 Create Account Page - Database Design Account Handling 1h	PPA-91 Create Account Page - Draft GUI Account Handling 3h			~~PPA-32~~ Learn MEAN Stack MEAN Stack Research 0h
PPA-100 Create Account Page - Middle Tier Logic Account Handling 2h				

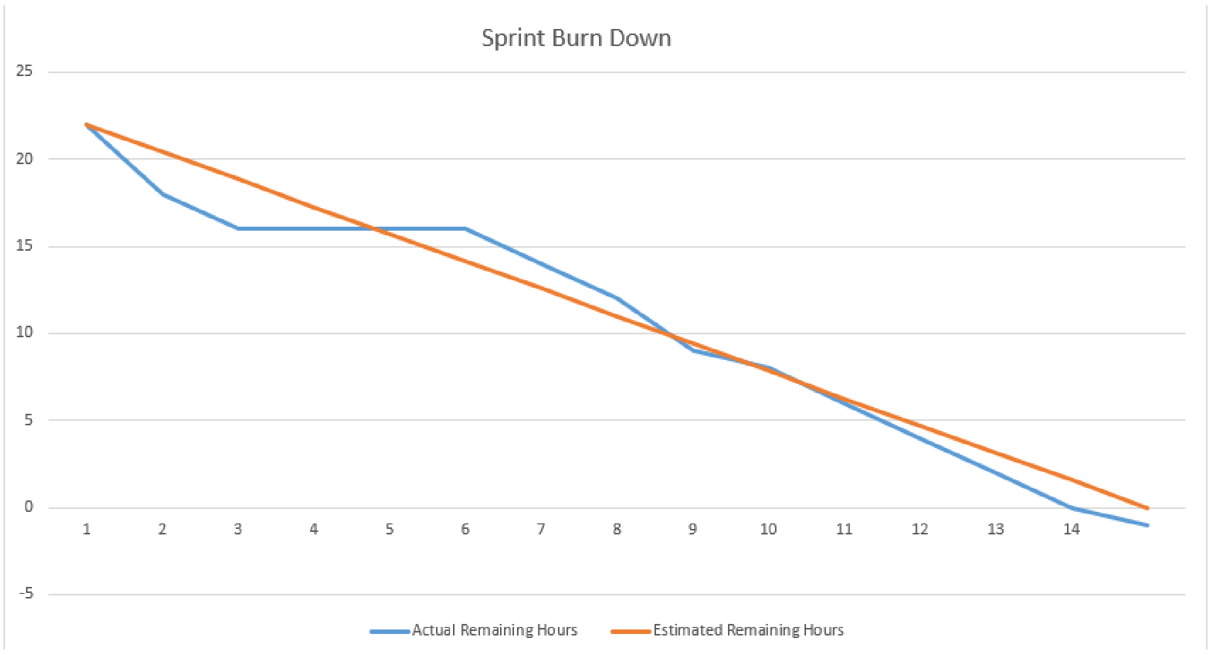
We had 5 standup meetings within this sprint and the minutes of those meetings can be found from the following link.

Standup Meeting Minutes:

https://bitbucket.org/Computing_Projects_SLIIT/2018_sd07/src/master/Documents/Standup%20Meeting%20Minutes/Sprint%203/Standup%20Meeting%20-%20Sprint%203.pdf

4.2 Burndown Chart

Estimate Time:	22 Hours
Actual Time:	23 Hours



4.3 Sprint Retrospective

Note: This is a continuing task for each sprint.

Estimate Time: 0.5 Hours

Actual Time: 0.5 Hours

Actual Time (this sprint): 0.5 Hours

4.4 Task Summary

Estimate Time:	22 Hours
Actual Time:	23 Hours

Completed 12 Task during this Sprint.

4.5 Time Management

#	Task ID	Task	Hours Estimated	Actual

1	PPA-	Detailed Account Page Designing	1	1
2	PPA-	Detailed Account Page Implementation	4	5
3	PPA-	Detailed Account Page - Database Design	1	1
4	PPA-	Detailed Account Page - Insert Method Calls	2	2
5	PPA-	Detailed Account Page Update Method Calls	2	2
6	PPA-	Detailed Account Page Delete Method Calls	2	2
7	PPA-	Detailed Account Page Validation Methods	2	2
8	PPA-	Detailed Account Page Transaction Table Insert Methods	2	2
9	PPA-	Detailed Account Page Transaction Table Update Methods	2	2
10	PPA-	Detailed Account Page Transaction Table Delete Methods	2	2
11	PPA-	Sprint 4 Planning	1	1
12	PPa-	Discuss design issues Sprint 3	1	1
	Total		**22**	**23**