

Starbucks Application

Team Project for CMPE 202: Software Systems Engineering(Spring 2019)

MEMBERS:

Arkil Thakkar(013825292)

Pranjal Sharma(013831688)

Rachit Saxena(012469626)

Shravani Pande(013849264)

Nehal Sharma(013849290)

Components Implemented

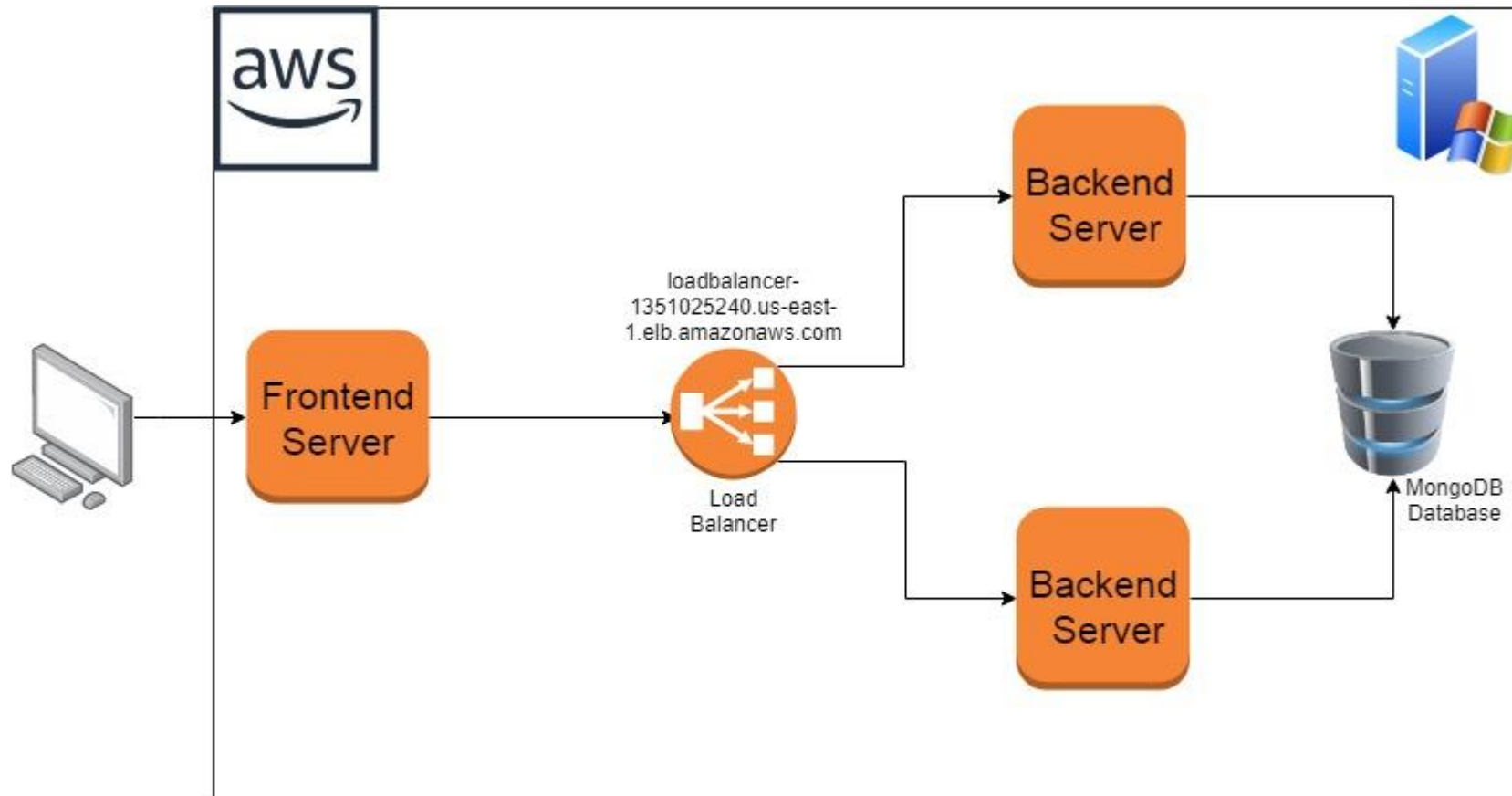
► Individual components:

- Authentication API: Implemented by Pranjal
- Add Card API: Implemented by Rachit
- Manage Order API: Implemented by Shravani
- Make Payment API: Implemented by Nehal
- Remove Card API: Implemented by Arkil

► Tasks completed for Extra Credit:

- Deploy API to AWS in an Auto Scaled EC2 Cluster with Load Balancer
- Implement Web "Front-End" Deployed to Heroku for Starbucks Payment Card management

System Architecture



Authentication API

ⓘ Not secure | ec2-18-204-130-151.compute-1.amazonaws.com:3000

Welcome to Starbucks

Enter Passcode

<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
1	2	3	
4	5	6	
7	8	9	
	0	X	

```
app.post('/pinValidation', (req, res)=>{
  const {pin} = req.body;
  console.log(pin);
  if(pin === '1234'){
    res.status(200).send(true);
  }
  else{
    res.status(202).send(false);
  }
});
```

Add Card API

▲ Not secure | ec2-18-204-130-151.compute-1.amazonaws.com:3000/addcard

ec2-18-204-130-151.compute-1.amazonaws.com:3000 says
Card added successfully

OK

123456789

234

Add Card

\$ 🏠 📋 ⚙️ 🔌

```
app.post('/addCard', (req, res) => {
  const {cardno} = req.body;
  const {cvv} = req.body;
  const card = new Card({
    _id: new mongoose.Types.ObjectId(),
    cardno: cardno,
    cvv: cvv,
    balance: 20
  });






  Card.find({cardno: cardno}).exec().then(result=>{
    if(result.length > 0){
      res.status(200).send(false);
    }
    else{
      card.save().then(result=>{
        console.log(result);
        res.status(200).send(true);
      })
    }
  });
});
```

Manage Order API

Not secure | ec2-18-204-130-151.compute-1.amazonaws.com:3000/orders

Welcome to Starbucks Orders

Card No	Time	
123456789	21:56	Cancel



```
app.post('/makeOrder', (req, res)=>{
  const cardno = req.body.cardno;
  const order = new Order({
    _id: new mongoose.Types.ObjectId(),
    cardno: cardno
  });

  order.save().then(result=>{
    console.log(result);
    res.status(200).json({
      message: "Order Placed Successfully"
    })
  })
})

app.post('/cancelOrder', (req, res)=>{
  const cardno = req.body.cardno;
  const orderid = req.body.orderid;
  Card.update({cardno: cardno}, {$inc: {balance: 1.5}}).exec().then(resultU=>{
    console.log(resultU);
    Order.remove({_id: orderid, cardno: cardno}).exec().then(result=>{
      res.status(200).json({
        message: "Order Cancelled"
      })
    })
  })
})
})
```

Make Payments API

Not secure | ec2-18-204-130-151.compute-1.amazonaws.com:3000/balance

Welcome to Starbucks

View Balances

Card No	Balance
123456789	17



Not secure | ec2-18-204-130-151.compute-1.amazonaws.com:3000/balance

Welcome to Starbucks

View Balances

Card No	Balance
123456789	15.5



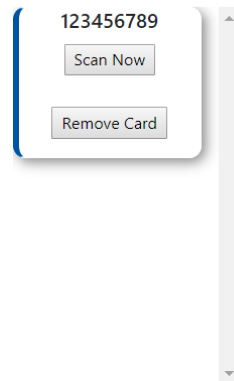
```
app.post('/minusBalance', (req, res)=>{
  const cardno = req.body.cardno;
  Card.find({cardno: cardno}).exec().then(resultB=>{
    console.log(resultB);
    if(resultB[0].balance >= 1.50){
      Card.update({cardno: cardno}, {$inc: {balance: -1.5}}).exec().then(resultU=>{
        res.status(200).send(true);
      })
    }
    else{
      res.status(200).send(false);
    }
  })
});
```

Remove Card API

Not secure | ec2-18-204-130-151.compute-1.amazonaws.com:3000/mycards

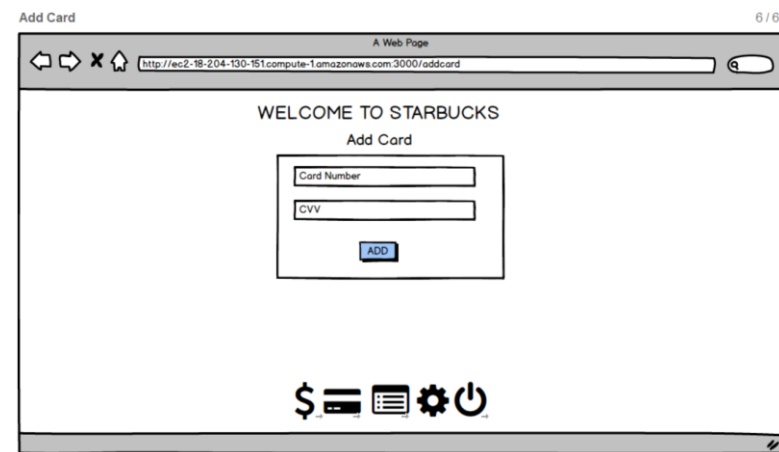
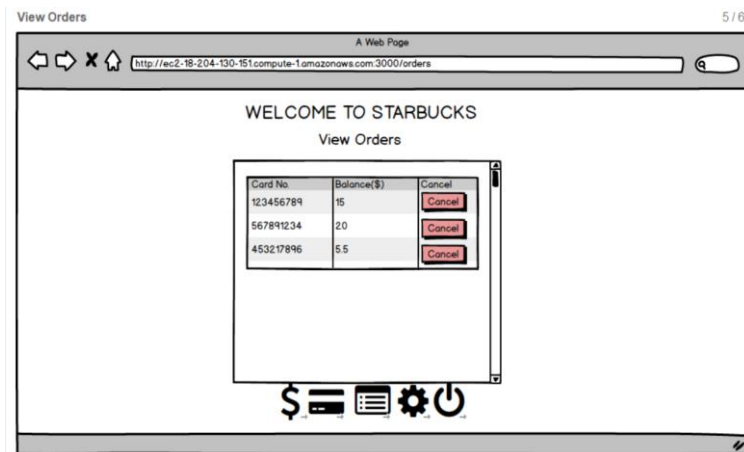
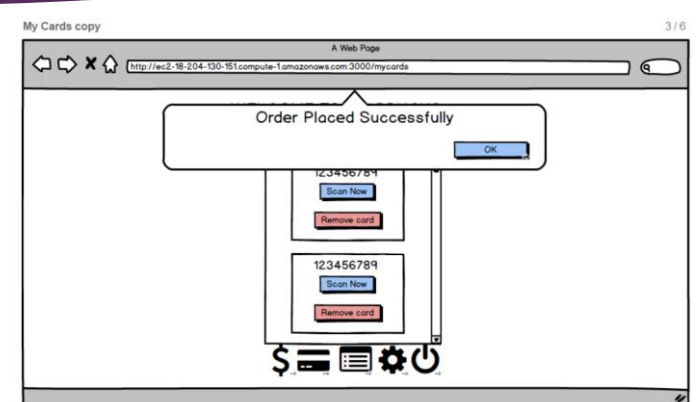
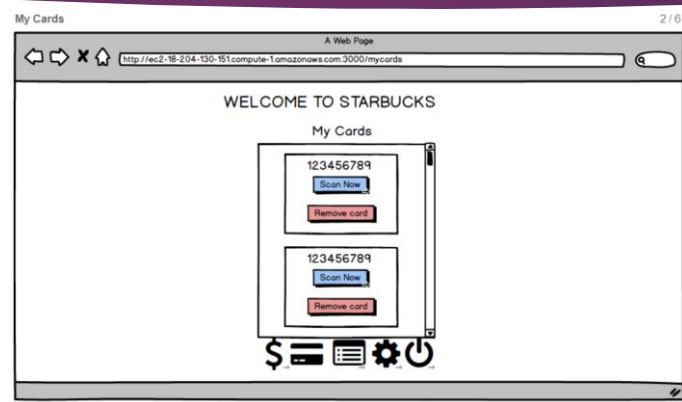
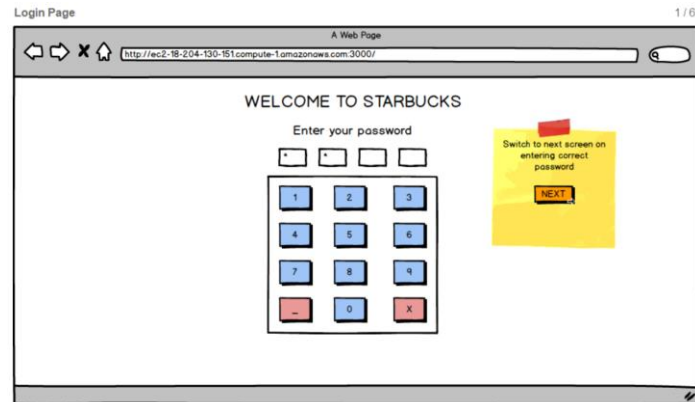
Welcome to Starbucks

My Cards

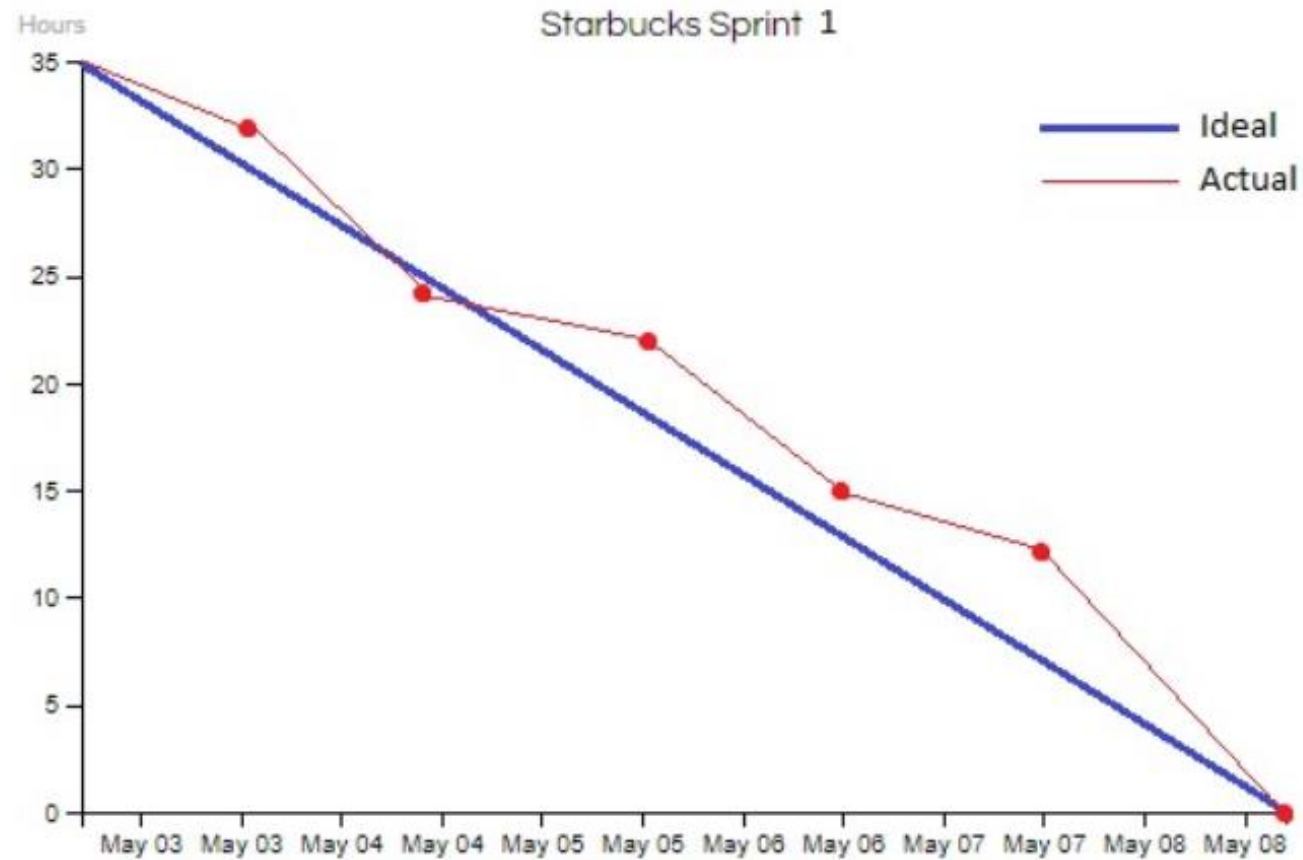


```
app.post('/removeCard', (req, res)=>{
  const cardno = req.body.cardno;
  Card.remove({cardno: cardno}).exec().then(result=>{
    console.log(result);
    res.status(200).json({
      message: "Card Removed Successfully"
    })
  })
});
```


UI Wireframes



Burndown Chart



Core XP Values

- ▶ **Simplicity : Taken care by Pranjal**

"Did the simplest thing that could possibly work" (DTSTTCPW principle) and implemented a new capability in the simplest possible way.

- ▶ **Communication : Taken care by Rachit**

Followed the *Sit Together* principle to keep the team members at a short distance from each other, to stimulate casual interactions and conversations.

- ▶ **Courage : Taken care by Nehal**

Diligently communicated and accepted feedback positively, was truthful about progress and estimates and made efforts to adapt to changes whenever they happen

- ▶ **Respect : Taken care by Shravani**

Inculcated respect for each other as a valued team member which led to driving a value system based on enthusiasm and mutual respect

- ▶ **Feedback : Taken care by Arkil**

Ensured every iteration commitment is taken seriously by delivering a working software.



Thank You!