**CMPE202 -  Starbucks Group Project-DRIFTERS**

**Instructor**: Gopinath Vinod

**Team Members:**

|  |  |
| --- | --- |
| **Student Name** | **Contribution** |
| Juilan Simon | **Add Card** |
| Paramdeep Saini | **User Profile and Authentication** |
| Sandhya Gadgoli | **Manage Order** |
| Vishwa Kambam | **Payments API** |

**Table Of Contents:**

* Introduction
* Design Decisions and Feature List
* Cloud Deployment
* Architecture Diagram
* Component Diagram
* Deployment Diagram
* Tools
* Links

**Introduction:**

* Requirement is to implement a full end-to-end Starbucks order management RESTful application.
* Develop REST API and use the business logic code as a JAR library.
* Deploy API to **AWS in an Auto Scaled EC2 Cluster** with Load Balancer
* Deploy API to AWS as Docker Containers in **Amazon Containers**
* Deploy API to AWS as Docker Containers in **Amazon EKS**
* Follow Scrum/Kanban methodologies for tracking the tasks and progress.
* Maintain a UML Class Diagram and a Sequence Diagram using professional tool like Astah UML.

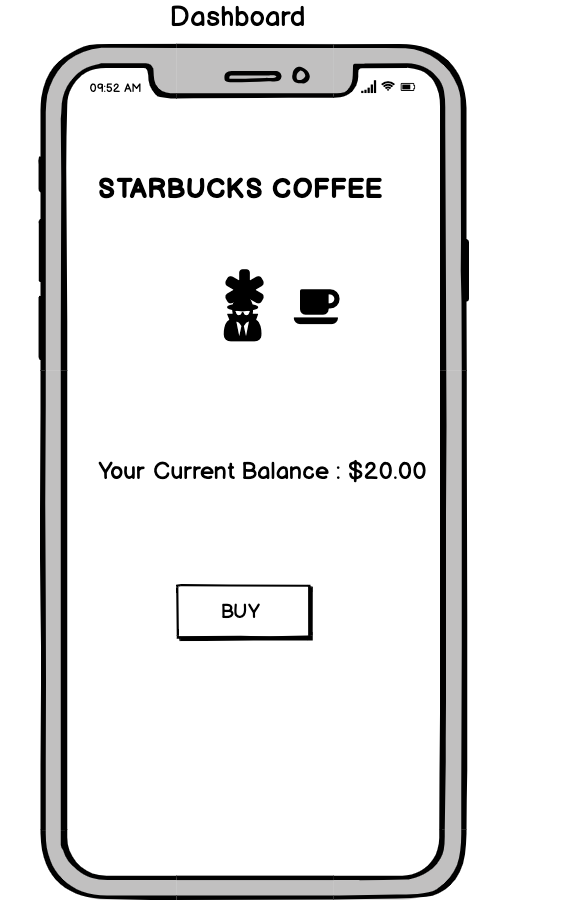
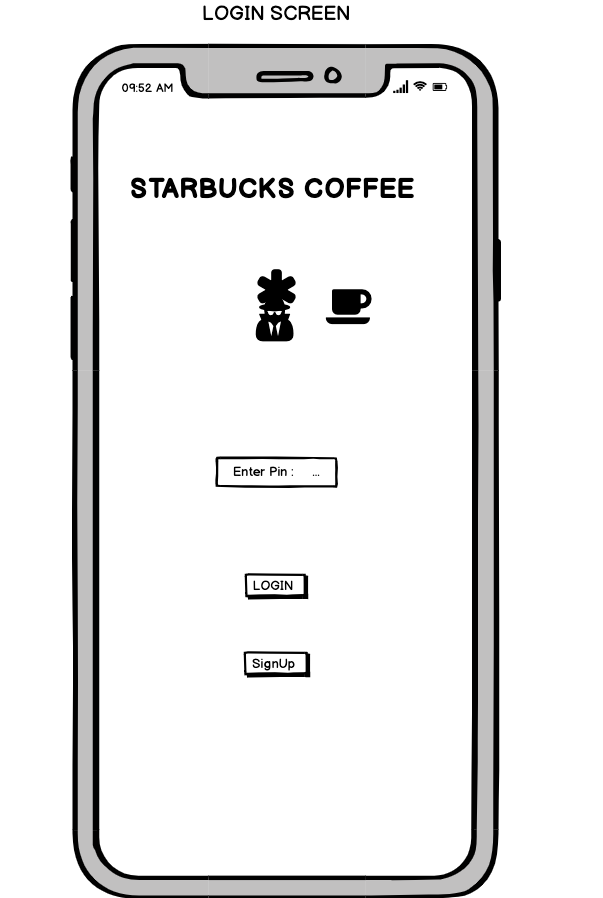
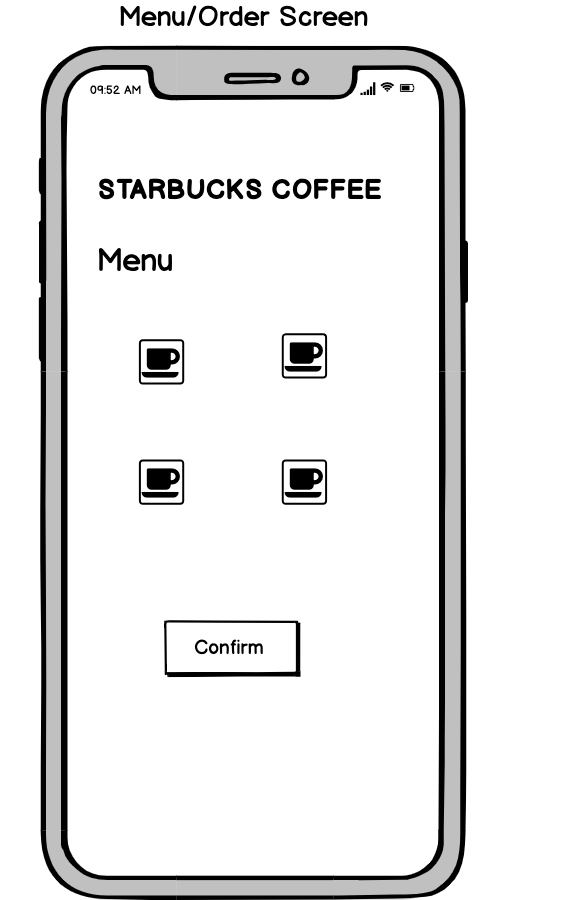
**Design decisions and feature List**

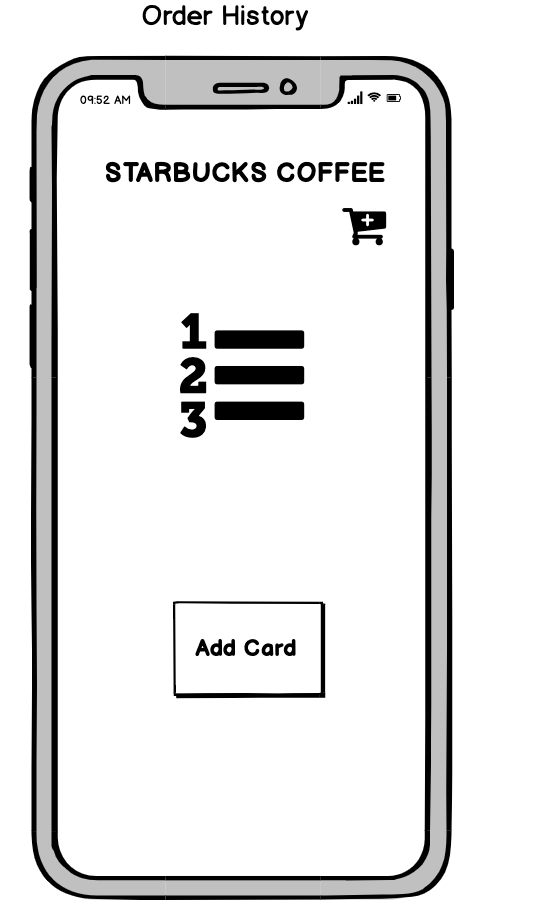
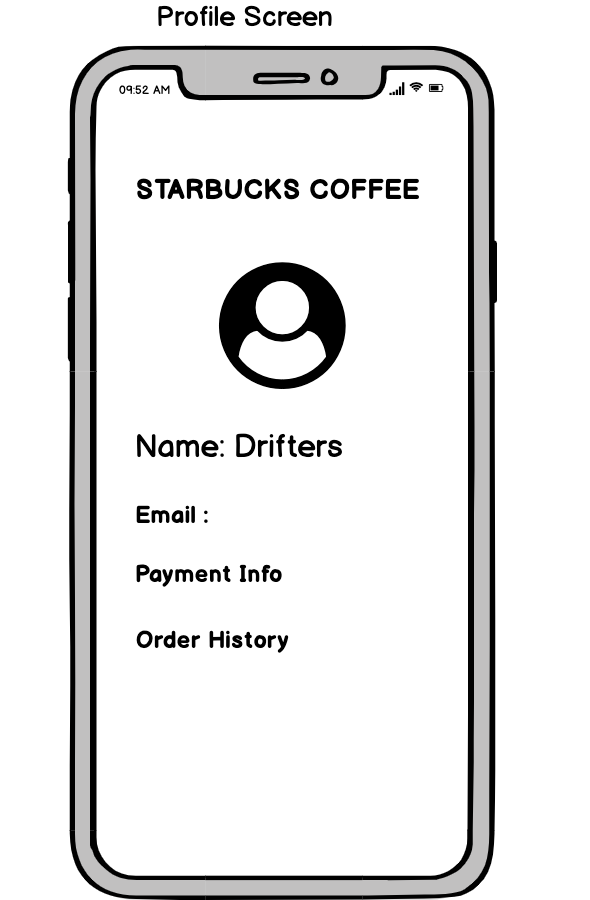
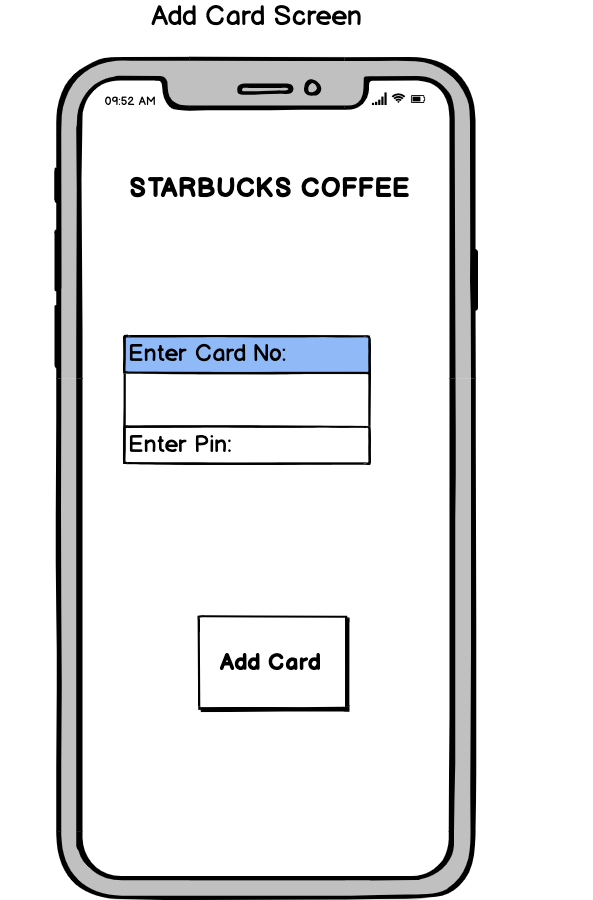
1. We decided to keep the application completely stateless. REST is an architectural style which is based on web-standards and the HTTP protocol. In a REST based architecture, you have a REST server which provides access to the resources. We have used gradle project in the Main starbucks-app , it’s JAR is imported checked in as an artifact.
2. Jersey REST controllers for exposing the API endpoints which is embedded with tomcat server for the Microservices based implementation. We have used our own Model controller and view controller and Service controller.
3. We wanted to authenticate all requests to the API's so we used JWT tokens for authentication.
4. We treat Starbucks.jar artifact as Model Controller since all the logic is implemented there.
5. We are needed output in json format hence added artifact id: jersey-media-json-jackson.
6. MySql for Data base
7. Maven for dependency management
8. Implemented the following components:
   * User profile and authentication API
   * Order Management API
   * Card Management API
   * Payment API
9. **Each customer can have only 1 card at a time he/she can update its balance or replace with a whole new card.**
10. **Each customer can have only 1 open order at a time and must pay for a new order.**
11. We are using singleton pattern and so if a new implementation of objects is present then minimal modification is required.

**Deployment:**

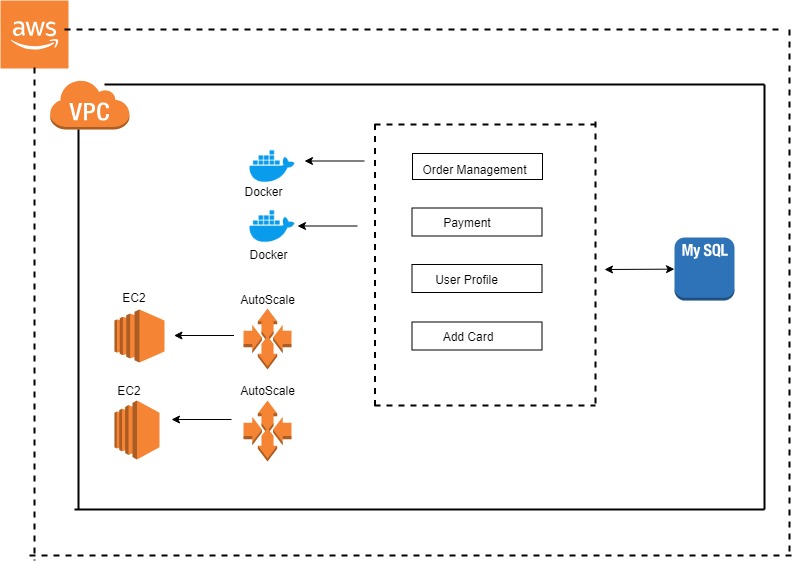
1. We have implemented Microservices architecture, all the APIs has been deployed to individual AWS cloud environments to have a proper decoupling between them.
2. Deployed Order Management and Card Management API to AWS Docker Containers using ECS.
3. Deployed Sign In / Sign Up and Payment API to AWS in an Auto Scaled EC2 Cluster with Load Balancer.

**Screen Mockup /WireFrames**

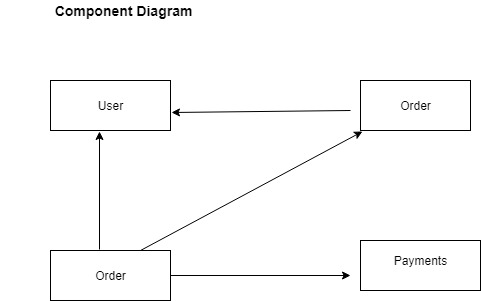
 

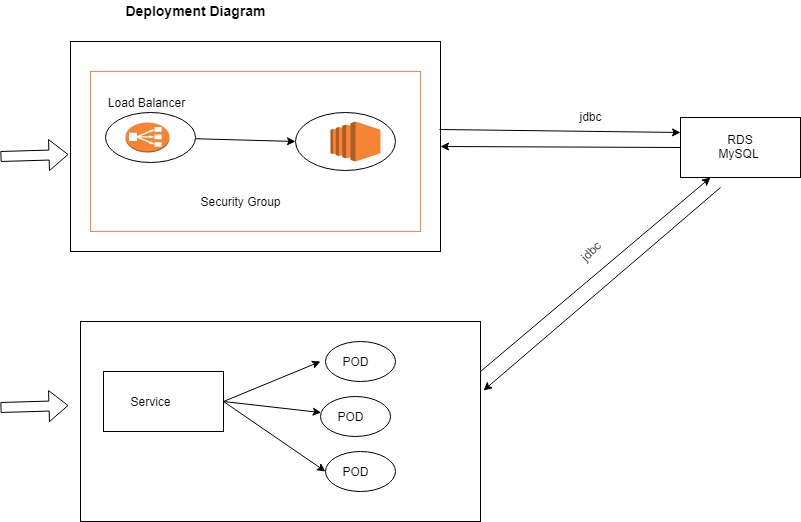
**Architecture Diagram**



**Component Diagram**



**Deployment Diagram**



**Tools**  
  
Eclipse

Gradle with jdk 1.8

Apache Tomcat

Postman  
Steps

1. Download repo psaini2018/JavaRest
2. Go in to downloaded folder
3. Use below commands to run:

gradle build

copy Starbucks.jar to /tmp folder

Run the Application on tomcat

Open Postman or browser to test APIs.(refer configurations for URI and payload)

**Links**

* + [Team's GitHub Repo](https://github.com/psaini2018/JavaRest)
  + [Project Journal](https://github.com/psaini2018/JavaRest/tree/master/Project%20Journal)
  + [Team XP core values](https://docs.google.com/document/d/1HQvyn7FHvx4vuNdeIVe_neRY-qHYOqJ12XRyjSrkF-8/edit#heading=h.mi8axact8wc1)
  + [Google Sprint Task Sheet](https://docs.google.com/spreadsheets/d/1mVZAOs_Rj2oF7O5L_x75E9ENvSp36ExLHa2L75i4AWU/edit#gid=0)