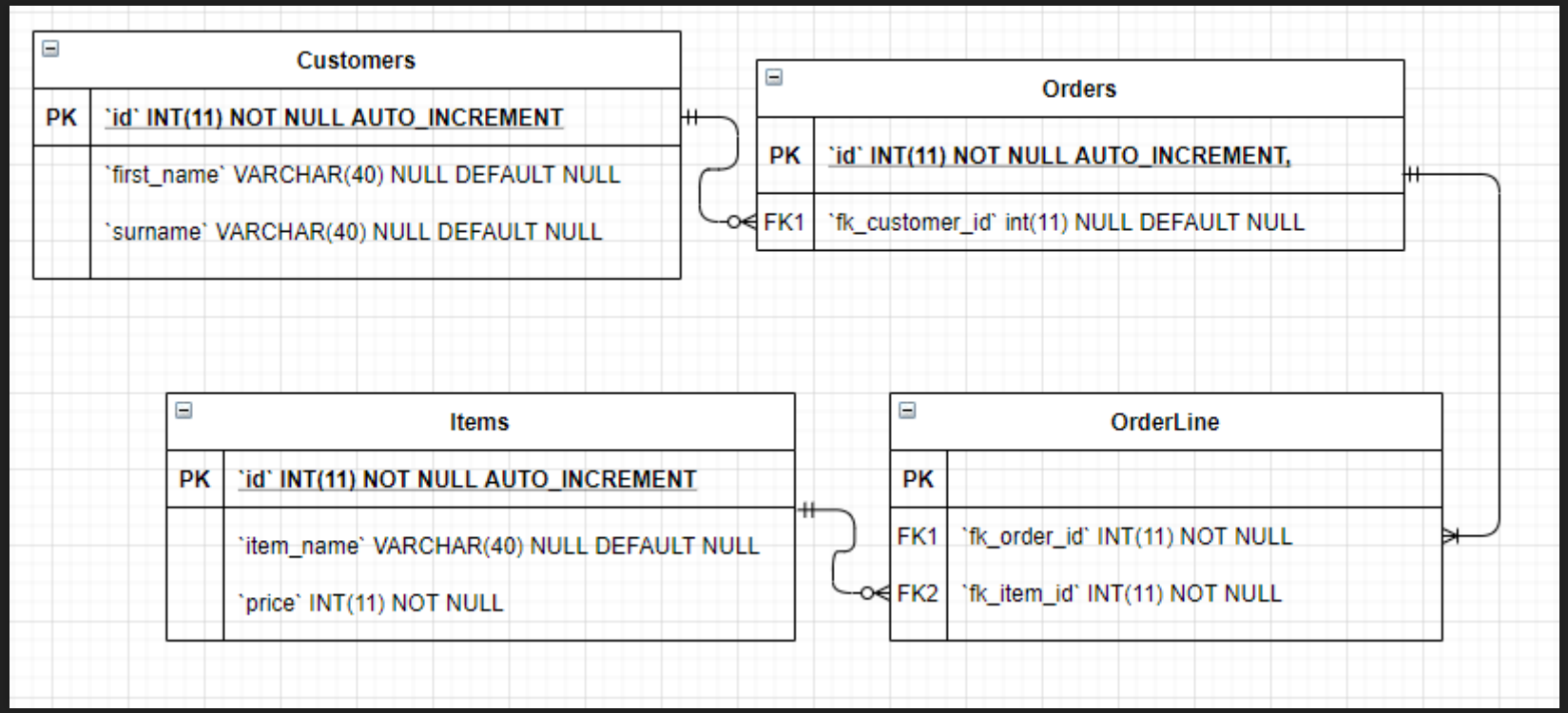
IMS STARTER QA

Benjamin Simon

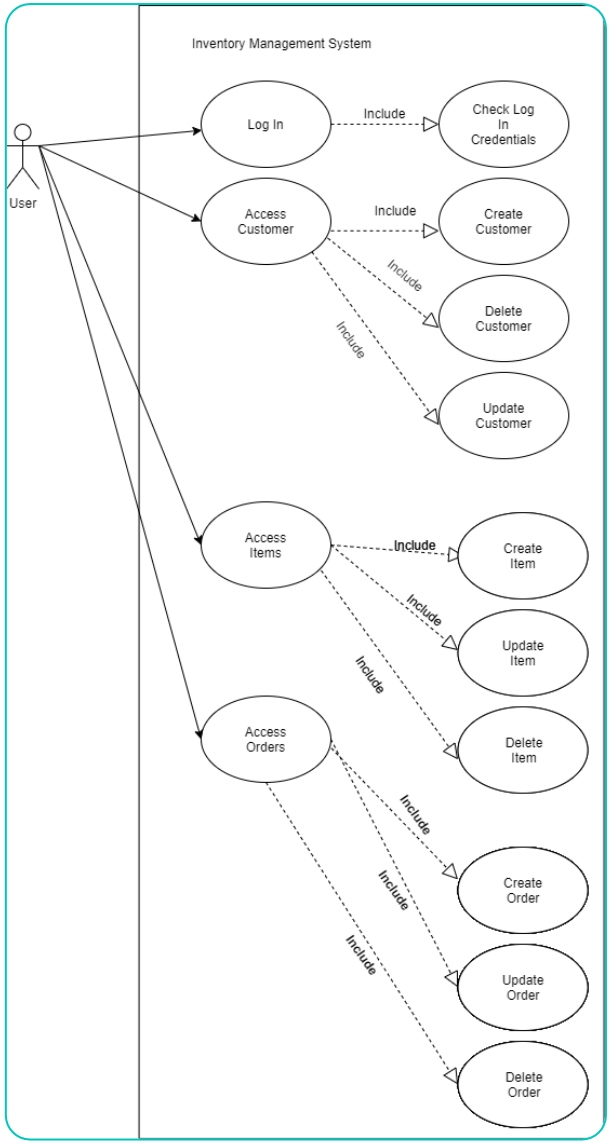
Introductions

Start of the IMS
Create ERD
Create UML
Make an Initial Risk Assessment
Make a MoSCoW
Set Up Jira and Github

ERD


Customers
PK
`id` INT(11) NOT NULL AUTO_INCREMENT
`first_name` VARCHAR(40) NULL DEFAULT NULL
`surname` VARCHAR(40) NULL DEFAULT NULL
Orders
PK
`id` INT(11) NOT NULL AUTO_INCREMENT,
FK1
`fk_customer_id` int(11) NULL DEFAULT NULL
Items
PK
`id` INT(11) NOT NULL AUTO_INCREMENT
`item_name` VARCHAR(40) NULL DEFAULT NULL
`price` INT(11) NOT NULL
OrderLine
PK
FK1
`fk_order_id` INT(11) NOT NULL
FK2
`fk_item_id` INT(11) NOT NULL

UML


Inventory Management System
User
Log In
Include
Check Log In Credentials
Access Customer
Include
Create Customer
Include
Delete Customer
Include
Update Customer
Access Items
Include
Create Item
Include
Update Item
Include
Delete Item
Access Orders
Include
Create Order
Include
Update Order
Include
Delete Order

Risk Assessment

Risk register 1 – Initial scores and actions

Ref	Risk Description	Cause	Risk Event	Likelihood (1-5)	Impact (1-5)	Risk Rating	Action
1	Lack of time	Improper use of time and lack of time management	Unable to complete assigned project and meeting all criteria.	3	4	10	Plan daily and weekly tasks. Set sprint and time goals.
2	Lack of experience	Not having proper work experience.	Not completing project at industry standards.	2	5	9	Ask questions when needed, complete given task and do further research.
3	Self-Management	Overwhelming and burnout.	Unable to finish project, loss of motivation and affecting health.	3	4	12	Take breaks, do what you can, set realistic tasks.
4	Worldwide Disruption	Covid-19.	Illness, change of routine, remote work.	5	3	9	Social distancing, make sure have all equipment's, join online calls.
5	Unreliable application	Not having enough test results.	Project is not fully reliable.	3	4	12	Test all methods, ask for help.

MoSCoW

Must haves:

- Code with a fully integrated version control System using GitHub and Git.
- A project management board that contains user stories, acceptance criteria and task that will need to complete the project.
- A risk assessment that outline all potential risk as well and risk that are faced during the production.
- A relational database that contains a customers, products, orders and orders, items tables.
- A functional “back-end” that follows the best industry practices and design, using all languages and materials covered during the course.
- A build of the application that fully functions.
- An industry standard of 80% test coverage.
- A well thought out presentation that covers everything about the project as well as materials covered during the training.

Should haves:

- Good use of comments throughout the code to get an understanding of what is happening.
- Constantly commit project to GitHub to have backups, that contains clear messages to know what has been committed.
- A clean and structured folder.

Could haves:

- A test coverage that goes above 95%.
- A daily dairy that covers everything that has been applied and created for the project.

Wont haves:

- Front end development.
- Images to display items, or even customers.
- Security and user log in credentials.

Consultant journey

Technologies
Eclipse IDE (JAVA)
GCP
SQL
Maven
Junit
Git and GitHub

Management
Jira
MoSCoW
Risk Assessment

IMS-Starter	84.3 %	3,516	655	4,171
src/main/java	74.7 %	1,934	655	2,589
com.qa.ims	0.0 %	0	171	171
com.qa.ims.controller	71.7 %	416	164	580
com.qa.ims.persistence.dao	97.3 %	1,062	30	1,092
com.qa.ims.persistence.domain	57.4 %	299	222	521
com.qa.ims.utils	69.8 %	157	68	225
src/test/java	100.0 %	1,582	0	1,582

CustomerDAOTest.java

```java
package com.qa.ims.persistence.dao;

import static org.junit.Assert.assertEquals;

public class CustomerDAOTest {

    private final CustomerDAO DAO = new CustomerDAO();

    @BeforeClass
    public static void init() {
        DBUtils.connect("root", "ben.simon.QAIMS");
    }

    @Before
    public void setup() {
        DBUtils.getInstance().init("src/test/resources/sql-schema.sql", "src/test/resources/s
    }

    @Test
    public void testCreate() {
        final Customer created = new Customer(3L, "chris", "perrins");
        assertEquals(created, DAO.create(created));
    }

    @Test
    public void testReadAll() {
        List<Customer> expected = new ArrayList<>();
        expected.add(new Customer(1L, "Jordan", "Harrison"));
        expected.add(new Customer(2L, "Jack", "Sparrow"));
        assertEquals(expected, DAO.readAll());
    }

    @Test
    public void testReadLatest() {
        assertEquals(new Customer(2L, "Jack", "Sparrow"), DAO.readLatest());
    }

    @Test
    public void testRead() {
        final long ID = 1L;
        assertEquals(new Customer(ID, "Jordan", "Harrison"), DAO.readCustomer(ID));
    }

}
```

CustomerDAO.java

```java
package com.qa.ims.persistence.dao;

import java.sql.Connection;

public class CustomerDAO implements Dao<Customer> {

    public static final Logger LOGGER = LogManager.getLogger();

    @Override
    public Customer modelFromResultSet(ResultSet resultSet) throws SQLException {
        Long id = resultSet.getLong("id");
        String firstName = resultSet.getString("first_name");
        String surname = resultSet.getString("surname");
        return new Customer(id, firstName, surname);
    }

    /**
     * Reads all customers from the database
     *
     * @return A list of customers
     */
    @Override
    public List<Customer> readAll() {
        try (Connection connection = DBUtils.getInstance().getConnection();
                Statement statement = connection.createStatement();
                ResultSet resultSet = statement.executeQuery("select * from customers");) {
            List<Customer> customers = new ArrayList<>();
            while (resultSet.next()) {
                customers.add(modelFromResultSet(resultSet));
            }
            return customers;
        } catch (SQLException e) {
            LOGGER.debug(e);
            LOGGER.error(e.getMessage());
        }
        return new ArrayList<>();
    }

    public Customer readLatest() {
        try (Connection connection = DBUtils.getInstance().getConnection();
                Statement statement = connection.createStatement();
                ResultSet resultSet = statement.executeQuery("SELECT * FROM customers ORDER
            resultSet.next();
```

Testing

User Stories

As a customer, I will like to see the total cost, so that I know how much I am paying.

~~JIBQ-3~~ BS

As a worker, I will like to add items to the database, so that customers know what products are available.

~~JIBQ-2~~ BS

As a customer, I will like to add my profile to the system, so that i can place orders.

~~JIBQ-1~~ BS

```
read
id:1 customer id:1 item_id:[id:1 item name:Black Ops price:20] total_price:20
```

```
create
Please enter the items name
SpiderMan
Please enter items price
60
Item created
What would you like to do with item:
CREATE: To save a new entity into the database
READ: To read an entity from the database
UPDATE: To change an entity already in the database
DELETE: To remove an entity from the database
RETURN: To return to domain selection
read
id:1 item name:Black Ops price:20
id:2 item name:SpiderMan price:60
```

```
create
Please enter a first name
Frank
Please enter a surname
Prank
Customer created
What would you like to do with customer:
CREATE: To save a new entity into the database
READ: To read an entity from the database
UPDATE: To change an entity already in the database
DELETE: To remove an entity from the database
RETURN: To return to domain selection
read
id:1 first name:Jordan surname:Harrison
id:2 first name:Jack surname:Sparrow
id:3 first name:Frank surname:Prank
```

Sprint Review

Completed
Created the UML, ERD and Risk Assessment.
Set Up a Jira and GitHub.
Have a fully functioning DAO and Controller
Reach 80% Test

Struggle
Deleting Customers when linked to orders.
Reimport initial project to IMS-Starter
Repository in a repository.

Sprint Retrospective

What Went Well
Time management
Testing
GCP
Eclipse Java Programming

What could be improved
Jira
GitHub

Conclusion

- Expanded knowledge on Java.
- Learned how to work ins SQL, GitHub and Jira.
- Learned how to test using Junit and Mockito.
- Best practice of Time Management.

ANY QUESTIONS?