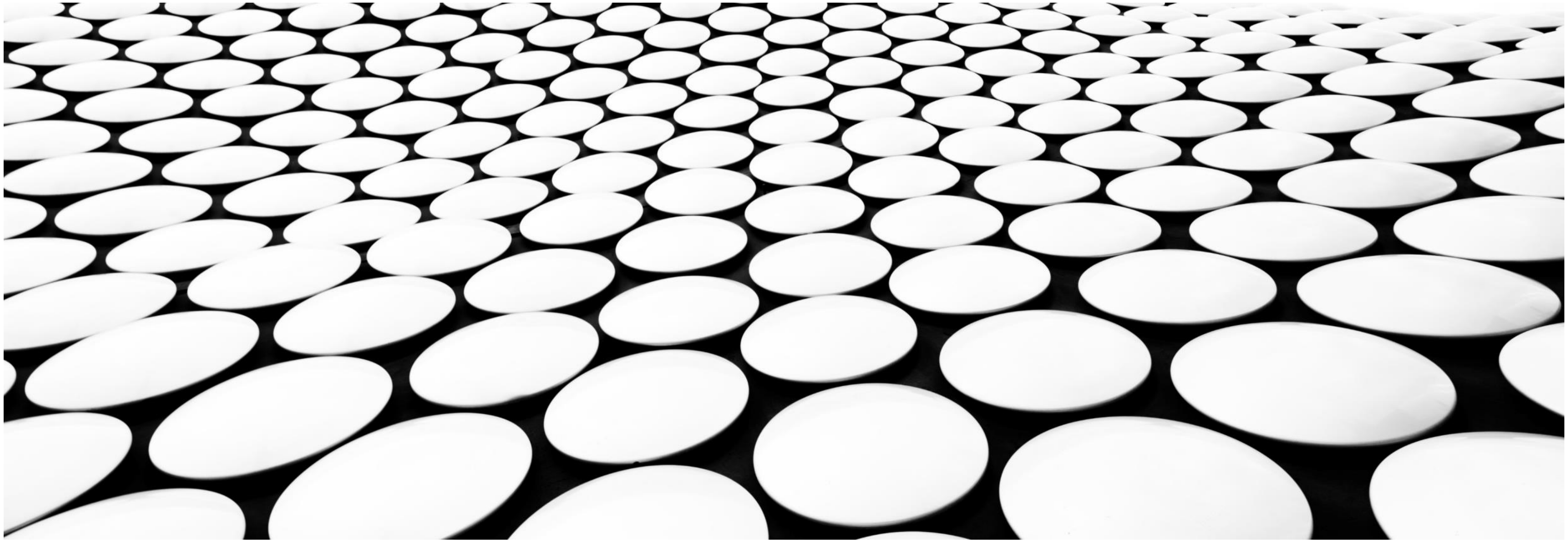
IMS PROJECT

22APRENABLE1

LADISLAV MUDRY



Introduction pt. 1

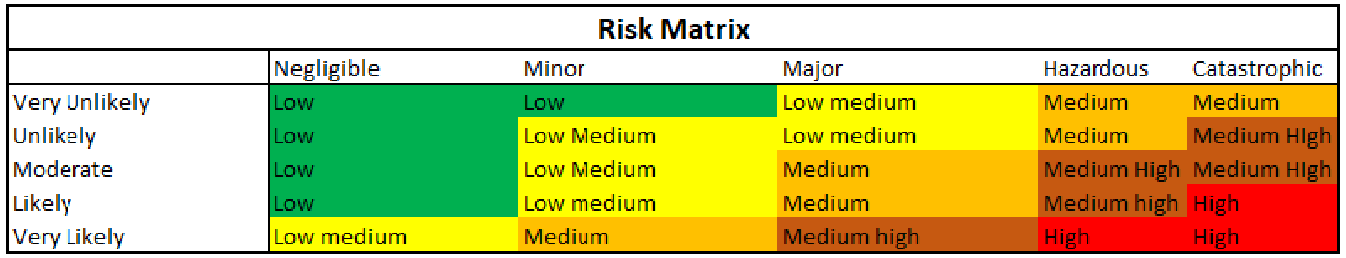
- Project‘s MVP deliverables:
  - Customers database with name (add, update, delete, view all)
  - Items database with name and value (add, update, delete, view all)
  - Orders database with customer and items (create, view all, delete)
  - Orders_items database (add item to order, delete item, calculate cost)
    - Additional db as MySQL does not support many to many relationships

  - Documentation
  - Git branching and uploading
    - Main/dev/features branching
  - Unit testing

Introduction pt. 2

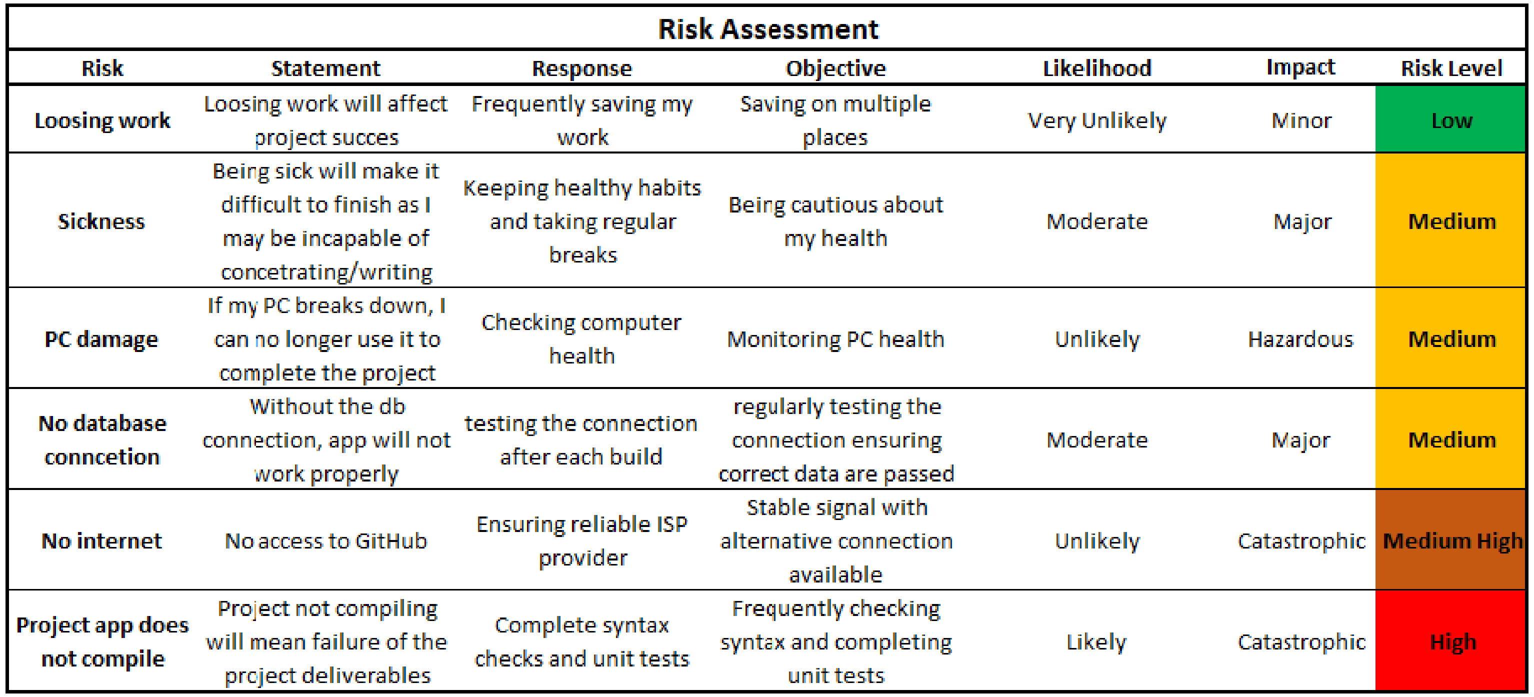
- Tools:
  - **Jira -** Kanban board for project management – including epics, user stories, tasks
  - **Git -** version control system
  - **GitHub -** source code management
  - **MySQL -** database for storing persistent data – JDBC connection to the project
  - **Eclipse** IDE - development environment with Maven, Junit and other dependencies
  - **Coding in Java**

Risk assessment and Matrix



Risk Matrix

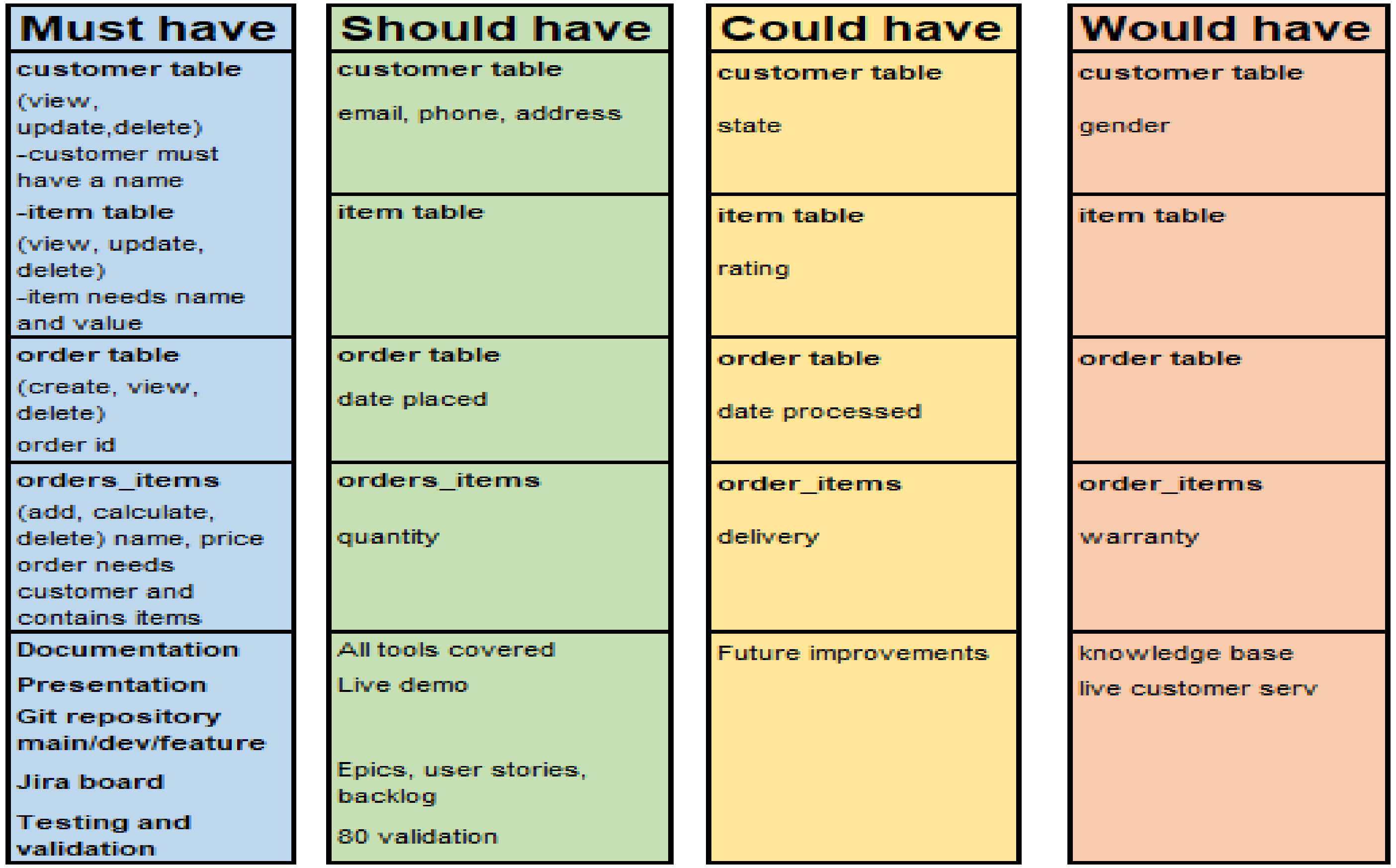
	Negligible	Minor	Major	Hazardous	Catastrophic
Very Unlikely	Low	Low	Low medium	Medium	Medium
Unlikely	Low	Low Medium	Low medium	Medium	Medium HIgh
Moderate	Low	Low Medium	Medium	Medium High	Medium HIgh
Likely	Low	Low medium	Medium	Medium high	High
Very Likely	Low medium	Medium	Medium high	High	High



Risk Assessment

Risk	Statement	Response	Objective	Likelihood	Impact	Risk Level
Loosing work	Loosing work will affect project succes	Frequently saving my work	Saving on multiple places	Very Unlikely	Minor	Low
Sickness	Being sick will make it difficult to finish as I may be incapable of concetrating/writing	Keeping healthy habits and taking regular breaks	Being cautious about my health	Moderate	Major	**Medium**
PC damage	If my PC breaks down, I can no longer use it to complete the project	Checking computer health	Monitoring PC health	Unlikely	Hazardous	**Medium**
No database conncetion	Without the db connection, app will not work properly	testing the connection after each build	regularly testing the connection ensuring correct data are passed	Moderate	Major	**Medium**
No internet	No access to GitHub	Ensuring reliable ISP provider	Stable signal with alternative connection available	Unlikely	Catastrophic	**Medium High**
Project app does not compile	Project not compiling will mean failure of the prcject deliverables	Complete syntax checks and unit tests	Frequently checking syntax and completing unit tests	Likely	Catastrophic	**High**

MoSCoW

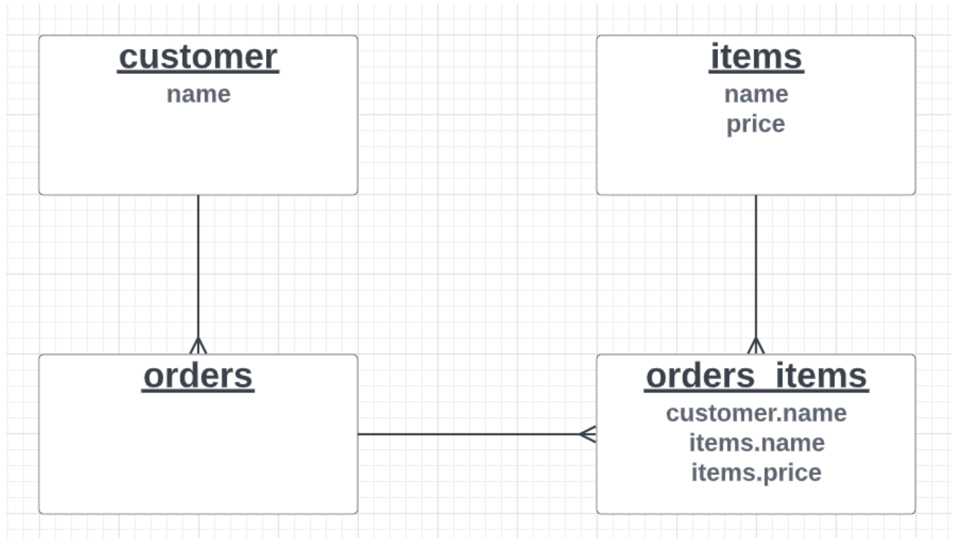


Must have	Should have	Could have	Would have
customer table (view, update,delete) -customer must have a name -item table (view, update, delete) -item needs name and value	customer table email, phone, address	customer table state	customer table gender
	item table	item table rating	item table
order table (create, view, delete) order id	order table date placed	order table date processed	order table
orders_items (add, calculate, delete) name, price order needs customer and contains items	orders_items quantity	order_items delivery	order_items warranty
Documentation Presentation Git repository main/dev/feature Jira board Testing and validation	All tools covered Live demo Epics, user stories, backlog 80 validation	Future improvements	knowledge base live customer serv

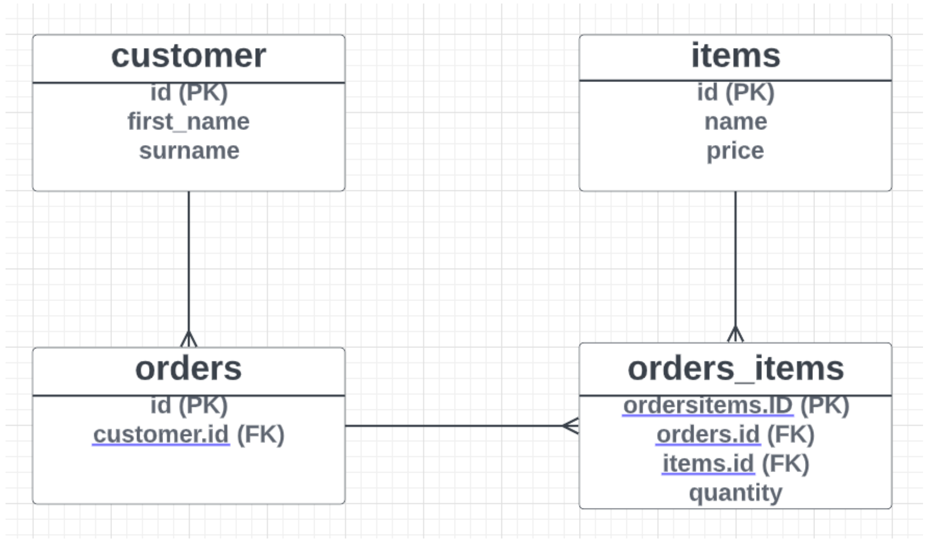
08.05.2022

ERD

- Before coding



- After coding

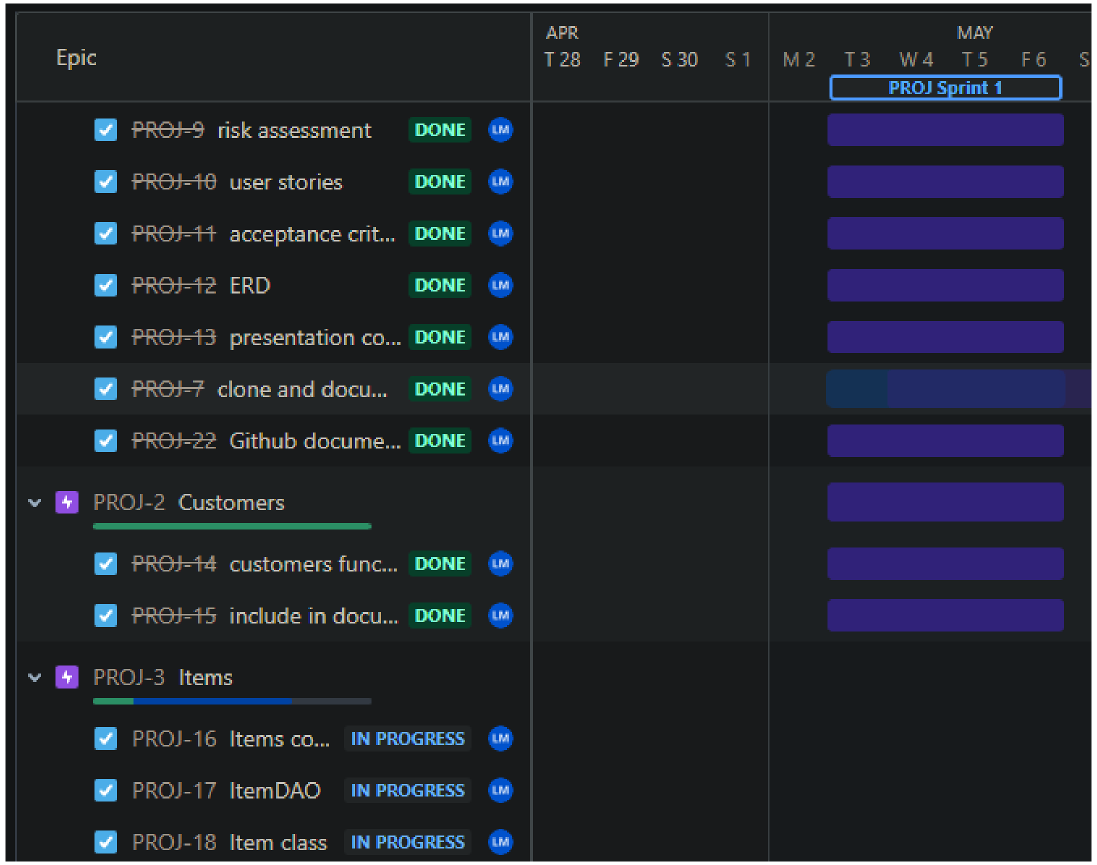


Sprints

- Sprint from Tuesday 10am to Friday 17.30pm
    - Divided into epics (for example item database)
        - Can be further divided into user stories (like a customers view and shop view, management view etc.)
            - Which is further divided into tasks (issues in Jira)
- Due to PC failure and unavailable/corrupt project files, extension was granted until Monday 9am, I created new Jira board but left original project dates

- Backlog – list of issues to be done in order to complete the project

Jira Kanban Board



Epic		
~~PROJ-9~~ risk assessment	DONE	LM
~~PROJ-10~~ user stories	DONE	LM
~~PROJ-11~~ acceptance crit...	DONE	LM
~~PROJ-12~~ ERD	DONE	LM
~~PROJ-13~~ presentation co...	DONE	LM
~~PROJ-7~~ clone and docu...	DONE	LM
~~PROJ-22~~ Github docume...	DONE	LM
PROJ-2 Customers		
~~PROJ-14~~ customers func...	DONE	LM
~~PROJ-15~~ include in docu...	DONE	LM
PROJ-3 Items		
PROJ-16 Items co...	IN PROGRESS	LM
PROJ-17 ItemDAO	IN PROGRESS	LM
PROJ-18 Item class	IN PROGRESS	LM

User Stories

- **As a user**, **I want to** access **customers** database **so that** I can view all customers and add, update, and delete individual customers.
- **As a user**, **I want to** access **items** in the database **so that** I can view all items and add, update, and delete individual items.
- **As a user**, **I want to** access **orders** in the database **so that** I can view all orders and add orders and delete individual orders.
- **As a user**, **I want to** add item to an order in the database **so that** I can calculate the cost of an order and delete individual item in the order.

Acceptance criteria (items)

- **Given,** the user has selected **items** in the system **and** wants to **CREATE** an item, **then** clicking the appropriate menu allows him to input item name and value into the system.
- **Given,** the user has selected **items** in the system **and** wants to **READ** all the items, **then** clicking the appropriate menu allows him to read all items with their names and values in the system.
- **Given,** the user has selected **items** in the system **and** wants to **UPDATE** an item, **then** clicking the appropriate menu allows him to input item name and value into the system.
- **Given,** the user has selected **items** in the system **and** wants to **DELETE** an item, **then** clicking the appropriate menu allows him to input item name and value into the system.

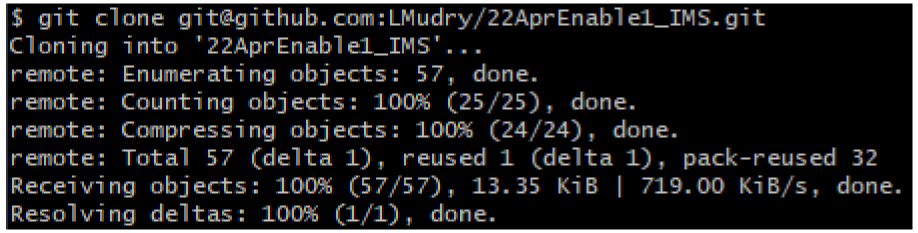
Acceptance criteria (orders)

- **Given,** the user has selected **orders** in the system **and** wants to **CREATE** an order, **then** clicking the appropriate menu allows him to input the order items and customer.
- **Given,** the user has selected **orders** in the system **and** wants to **READ** all the orders, **then** clicking the appropriate menu allows him to read all items in orders and the ordering customer.
- **Given,** the user has selected **orders** in the system **and** wants to **UPDATE** an item in the order, **then** clicking the appropriate menu allows him to update the selected item int the system.
- **Given,** the user has selected **orders** in the system **and** wants to **DELETE** an item in the order, **then** clicking the appropriate menu allows him to delete the selected item from the order.
- **Given,** the user has selected **orders** in the system **and** wants to **CALCULATE** the order totals, **then** clicking the appropriate menu allows him to calculate the sum if values of the ordered items.

GitHub

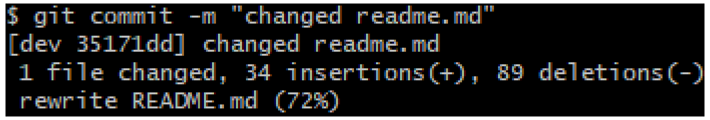
Forked the project branch and created a new dev branch

Cloned the branch to my local repository



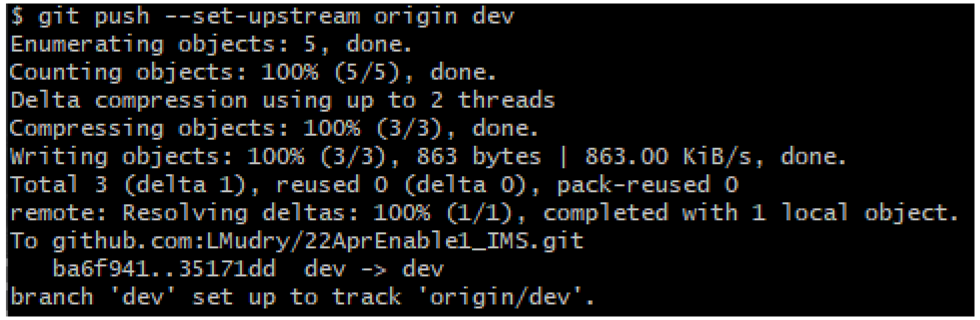
```
$ git clone git@github.com:LMudry/22AprEnable1_IMS.git
Cloning into '22AprEnable1_IMS'...
remote: Enumerating objects: 57, done.
remote: Counting objects: 100% (25/25), done.
remote: Compressing objects: 100% (24/24), done.
remote: Total 57 (delta 1), reused 1 (delta 1), pack-reused 32
Receiving objects: 100% (57/57), 13.35 KiB | 719.00 KiB/s, done.
Resolving deltas: 100% (1/1), done.
```

commited the changes in readme.md



```
$ git commit -m "changed readme.md"
[dev 35171dd] changed readme.md
 1 file changed, 34 insertions(+), 89 deletions(-)
 rewrite README.md (72%)
```

pushed to remote dev branch on GitHub



```
$ git push --set-upstream origin dev
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 2 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 863 bytes | 863.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To github.com:LMudry/22AprEnable1_IMS.git
   ba6f941..35171dd  dev -> dev
branch 'dev' set up to track 'origin/dev'.
```

SQL items table

```
-- creating table items
CREATE TABLE IF NOT EXISTS `ims`.`items` (
    `id` INT(11) NOT NULL AUTO_INCREMENT,
    `itemName` VARCHAR(40) DEFAULT NULL,
    `itemPrice` VARCHAR(10) DEFAULT NULL,
    PRIMARY KEY (`id`)
);
```

SQL orders table

```
-- creating table orders
CREATE TABLE IF NOT EXISTS `ims`.`orders` (
    `id` INT(11) NOT NULL AUTO_INCREMENT,
     `customers_id` INT NULL,
    PRIMARY KEY (`id`),
    CONSTRAINT `customers_id`
    FOREIGN KEY (`customers_id`)
    REFERENCES `test`.`customers` (`id`)
);
```

SQL order_items table

```
-- -- creating table order_items
CREATE TABLE `ims`.`orders_items` (
    `idordders_items` INT NOT NULL,
    `orders_id` INT NULL,
    `items_id` INT NULL,
    PRIMARY KEY (`idordders_items`),
    CONSTRAINT `orders_id`
    FOREIGN KEY (`orders_id`)
    REFERENCES `test`.`orders` (`id`),
    CONSTRAINT `items_id`
    FOREIGN KEY (`items_id`)
    REFERENCES `test`.`items` (`id`)
);
```

Creating an item CRUD

```
/**
 * Creating an item by taking in user input for logger in the system
 */
@Override
public Item create() {
LOGGER.info("Please enter item name");
String itemName = utils.getString();
LOGGER.info("Please enter a price");
String itemPrice = utils.getString();
Item item = itemDAO.create(new Item(itemName, itemPrice));
LOGGER.info("Item created");
return item;
}
```

Reading all items CRUD

```
/**
 * Reading all items to the logger in the system
 */
@Override
public List<Item> readAll() {
List<Item> items = itemDAO.readAll();
for (Item item : items) {
LOGGER.info(item);
}
return items;
}
```

Updating an item CRUD

```
/**
 * Updating an existing item by taking in user input in the system
 */
@Override
public Item update() {
LOGGER.info("Please enter the id of the item you would like to update");
Long id = utils.getLong();
LOGGER.info("Please enter an item name");
String itemName = utils.getString();
LOGGER.info("Please enter a price");
String itemPrice = utils.getString();
Item item = itemDAO.update(new Item(id, itemName, itemPrice));
LOGGER.info("Item Updated");
return item;
}
```

Deleting an item CRUD

```
/**
 * Deleting an existing item by the id of the item
 */
@Override
public int delete() {
LOGGER.info("Please enter the id of the item you would like to delete");
Long id = utils.getLong();
return itemDAO.delete(id);
}
```

Unit Testing item create()

```
// @RunWith(MockitoJUnitRunner.class)
@Test
public void testCreate() {
final String I_NAME = "bread", I_PRICE = "1.00";
final Item created = new Item(I_NAME, I_PRICE);

Mockito.when(utils.getString()).thenReturn(I_NAME, I_PRICE);
Mockito.when(dao.create(created)).thenReturn(created);

assertEquals(created, controller.create());

Mockito.verify(utils, Mockito.times(2)).getString();
Mockito.verify(dao, Mockito.times(1)).create(created);
}
```

Unit Testing item update()

```java
@Test
public void testUpdate() {
Item updated = new Item(1L, "Milk", "1.01");

Mockito.when(this.utils.getLong()).thenReturn(1L);
Mockito.when(this.utils.getString()).thenReturn(updated.getItemName(), updated.getItemPrice());
Mockito.when(this.dao.update(updated)).thenReturn(updated);

assertEquals(updated, this.controller.update());

Mockito.verify(this.utils, Mockito.times(1)).getLong();
Mockito.verify(this.utils, Mockito.times(2)).getString();
Mockito.verify(this.dao, Mockito.times(1)).update(updated);
}
```

Unit Testing item delete()

```
@Test
public void testDelete() {
final long ID = 1L;

Mockito.when(utils.getLong()).thenReturn(ID);
Mockito.when(dao.delete(ID)).thenReturn(1);

assertEquals(1L, this.controller.delete());

Mockito.verify(utils, Mockito.times(1)).getLong();
Mockito.verify(dao, Mockito.times(1)).delete(ID);
}
```

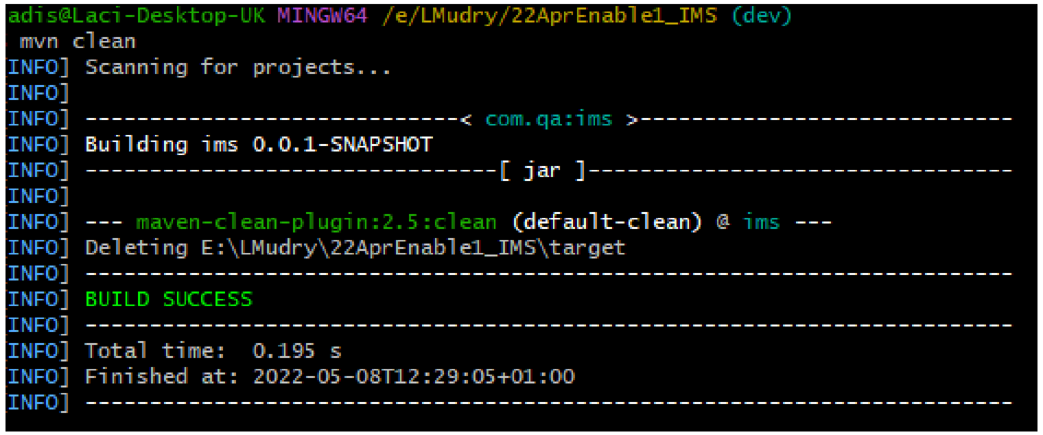
Encountered Issues

- **PC failure (corrupted, lost work) – extension granted for weekend (no help available for ad-hoc issues)**
- SSH issues with GitHub – resolved, created and uploaded new ssh
- Database connection issues – resolved, time zone and authentication issues
- Written code errors – single point of failure, difficult to identify a problem with your own code (pair programming advantage)
- Equalsverifier error message preventing creation of a maven build – not resolved, “mvn clean” successful but not package created due to errors – unable to create version builds, fatJar!
- Tried to version and build after cloning the project, updating or changing files, with no success.

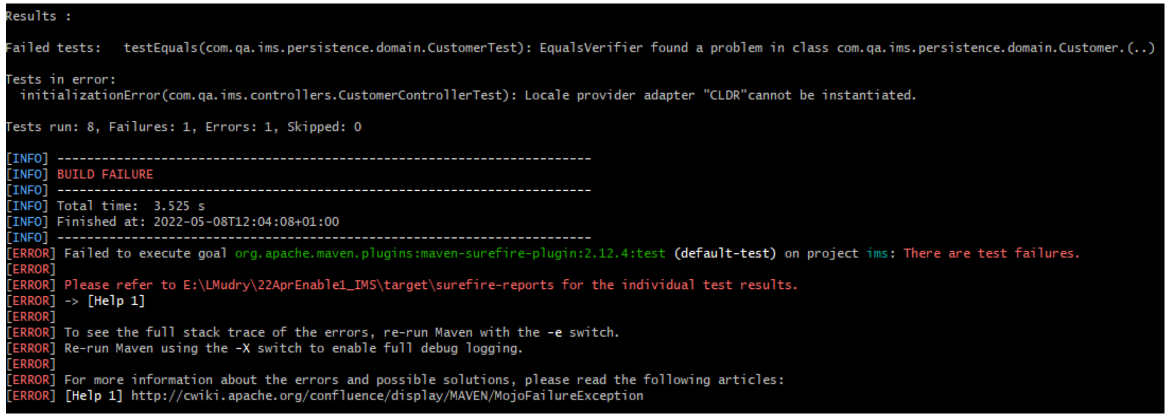
Failed packaging

Mvn clean – build success

Mvn package – build failure



```
adis@Laci-Desktop-UK MINGW64 /e/LMudry/22AprEnable1_IMS (dev)
 mvn clean
INFO] Scanning for projects...
INFO]
INFO] -----------------------------< com.qa:ims >------------------------------
INFO] Building ims 0.0.1-SNAPSHOT
INFO] --------------------------------[ jar ]---------------------------------
INFO]
INFO] --- maven-clean-plugin:2.5:clean (default-clean) @ ims ---
INFO] Deleting E:\LMudry\22AprEnable1_IMS\target
INFO] ------------------------------------------------------------------------
INFO] BUILD SUCCESS
INFO] ------------------------------------------------------------------------
INFO] Total time:  0.195 s
INFO] Finished at: 2022-05-08T12:29:05+01:00
INFO] ------------------------------------------------------------------------
```



```
Results :

Failed tests:   testEquals(com.qa.ims.persistence.domain.CustomerTest): EqualsVerifier found a problem in class com.qa.ims.persistence.domain.Customer.(..)

Tests in error:
  initializationError(com.qa.ims.controllers.CustomerControllerTest): Locale provider adapter "CLDR"cannot be instantiated.

Tests run: 8, Failures: 1, Errors: 1, Skipped: 0

[INFO] ------------------------------------------------------------------------
[INFO] BUILD FAILURE
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  3.525 s
[INFO] Finished at: 2022-05-08T12:04:08+01:00
[INFO] ------------------------------------------------------------------------
[ERROR] Failed to execute goal org.apache.maven.plugins:maven-surefire-plugin:2.12.4:test (default-test) on project ims: There are test failures.
[ERROR]
[ERROR] Please refer to E:\LMudry\22AprEnable1_IMS\target\surefire-reports for the individual test results.
[ERROR] -> [Help 1]
[ERROR]
[ERROR] To see the full stack trace of the errors, re-run Maven with the -e switch.
[ERROR] Re-run Maven using the -X switch to enable full debug logging.
[ERROR]
[ERROR] For more information about the errors and possible solutions, please read the following articles:
[ERROR] [Help 1] http://cwiki.apache.org/confluence/display/MAVEN/MojoFailureException
```

Sprint Review

- Project not completed to 100% of specification
  - IT issues, lack of time (no help available for weekend)
  - Approach: attempted most tasks even if not successfully completing versioning and builds
- missing some features
  - (calculating order totals)
- Encountered compilation and build errors, unable to create fatJar

Project Retrospective

- Planned as a real-life scenario with 4 days to complete the project
- IT issues – lost, corrupted resources
- Unable to make builds and fatJar – errors during packaging
- Went well
  - Gain understanding and hands-on experience with many tools (most of which I never used before)
  - A big step forward from simple exercises to real-life scenarios and requirements
  - Finding out how the project files work together to achieve the desired functionality
- Needs improvement
  - Git branching
  - Coding (Java)
  - Junit testing

Developer Journey

- Learned a range of new tools
  - GitHub, Java, Maven, Junit, working in Eclipse IDE
- Understand the basics of project management and what it involves
  - Jira boards, agile, pair programming
- Identified areas of improvement
  - Git, Java
- Gained more confidence using CLI rather then GUI

Conclusion

- Get more hands-on experience with tools and techniques learned with QA Community
- Motivated to be involved in projects to deepen my understandings of all the technologies working together
- **Looking forward to try and learn new tools/technologies and get involved in more projects to improve**

The End

- **Thank you for your time and opportunity to be part of QA Community**

- **Any questions ???**