**Final report**

The final report must contain an introduction and a conclusion. It must also have the following sections within the content body:

* · Project Management
  + The final report must contain a section on project management. Here you draw together the details of the work carried out and you analyse your team project management. Simply producing a glowing report of how well the group worked together is not sufficient; it is much better to recognize and evaluate any difficulties that have arisen and to demonstrate that you can reflect critically on your own work. It is an unusual project that does not encounter any problems.
  + Prior to writing the final report you should have a project closure meeting in which you review the project and the lessons learnt. Use the details from this meeting in the project management section of your final report.
  + The final report should also compare the plans you made at the start of the project and what had to change as you went along. In particular you should review your risk analysis and in the light of your experiences, comment on the appropriateness of your own evaluation. For example, did you identify appropriate risks, were your contingency plans appropriate?

* · Requirements Analysis and Design
  + A list of functional and non-functional requirements.

I'm pretty sure we have this list and won't be too hard to add

* + Analysis and design
    - A use case model (use case diagram(s) with some basic descriptions of use cases) for the final version of the software you have produced.
    - User stories with acceptance tests consistent with the use cases identified in the use case model.
    - User requirements list (product backlog)
    - Select FOUR use cases which will be demonstrated and produce design level sequence diagram, state machines and class diagrams. (A lot of diagram content needed here will need to fully crack on with this as soon as possible.)

* + - Evaluation of how your design changed during implementation.

Here we can talk about the fact we were very business orientated and were very ambitious with our functionality at the start

During the early stages of development we were ambitious with our early designs for the program. we had initially hoped to implement large amounts of functionality to enhance both the appearance and

* + - The analysis and design will be worth 10% of the final grade and the evaluation 5%.
* · Database
  + Your final report submissions must include a database diagram generated by Oracle SQL Developer, appropriate descriptions of your database objects (tables, views, sequences, database triggers, stored programs (procedures, functions)) and your SQL CREATE statements which should adhere to the naming and presentation conventions introduced in ISAD251.
  + Submit your normalized tables and compare your Schema with your Class Diagram. How does the final schema look like?
* · Security
  + Your final report should thoroughly explain what appropriate measurements you were able to implement and why. You should discuss how they improve the quality of the system. You should also discuss those aspects you feel are appropriate to the system but you were unable to implement because of time. Extra marks will be given if you pentest your application with some automated tools and are able to illustrate that it is robust.
* · Software Engineering
  + In the group report you are required to submit the following:
  + A UML class diagram for the application as outlined in section 5.2 on requirements. Your software engineering section must therefore provide an evaluation detailing any shortcomings, good points, and assumptions made.
  + The quality of your design will be evaluated against common OO quality guidelines for object-oriented systems such as Craig Larman’s GRASP (General Responsibility Assignment Software Patterns) guidelines and the use of established software patterns.
* · HCI
  + Your performance during the usability study itself will not be explicitly marked, but you are required to write a report on the user feedback obtained during the usability study and include this in your final report. The requirements for this section are as follows:
  + A progressive list (deriving from all user evaluations) of prioritized essential and desirable changes required. This list should be the result of synthesizing the data gathered during the usability evaluations of your product. In addition, you will also know of weaknesses in your product.
  + Each list must be prioritized in order of importance. The priority of items will also be assessed.
  + The HCI element will be marked on clarity. Note that diagrams may be added at the end of the document – these must be numbered Figure 1, 2 etc and referred to as such as in the main text.
  + The usability report must be a thorough critical evaluation of the usability of your product within the scope of this project. Please do not comment on features outside of this scope. Your report should also specify the strengths and weaknesses of your product and it should also report on user quotations, user preferences and the ease with which users were able to achieve their goals in task scenarios.

**Introduction**

For this project we were asked to create a complete system for a small pizza company. This includes functionality from ordering online, cooking all the way up to delivery.   
To split this up three main items were created.

First off we have the website using glassfish, css, html and the xserve server hosted by the uni.  
  
Then we have the admin/cooking program which is a two in one job. developed in netbeans using java and json/xml.

then we have the delivery mobile app developed in swift for iOS using xcode.

**Generating initial user-requirements**

* Client meetings

Prior to meeting the client we opted to have a group meeting to discuss what we felt key requirements for the various aspects of the project. These user requirements were primarily based on the initial brief that we were issued, however we made a point of investigating similar pre-existing systems and examining their functionality in order to develop any additional user requirements that may not have been initially apparent to us but would improve the usability of the system.

Following this we had a group meeting where we developed a list of questions that we wished to ask the client(s) in regard to what functionality they desired from the program. We did however make a point of only trying to establish what the customer wanted in regard to the core functionality of the project, this was because we were of the opinion that at this point we did not want to oversell the project by discussing additional features that we had conceptualised but were unsure whether we would actually be able to implement later in development.(maybe go into actual detail about what the questions were) (mention how we recorded the data for later use)

While all of our group were present during the initial client meeting we had decided prior that only one person would ask the client the prepared questions, while the rest of us would observe and record their response. More specifically those of us who were not directly communicating with the client were noting their responses on prepared question sheets, we felt that this approach to recording data would serve multiple purposes, firstly multiple people recording the same response would mean that there was a significantly diminished likelihood that any important information would fail to be recorded, multiple people recording responses also meant that we created multiple copies of the information, as opposed to making a single copy, mitigating the chance that critical data would be lost.

As a final measure we also made and audio recording of the entire session to try and eliminate any human error during the recording process, ensuring beforehand that the client gave full informed consent to be being recorded.

We did this so as to not overwhelm and intimidate the client since there would be several and only one of them. The person who actually engaged with the client was selected prior to the meeting based on their perceived ability by the group to best communicate with the client based on things such as a clear manner of speech and their general demeanour.

Following the initial client meeting we met together to analyse the client's responses to our questions and use these to generate a list of client defined user requirements. We then compared these with the user requirements that we had generated independently and added any requirements that we didn’t have to our list. After this we then prioritised our requirements in order of their importance, we referred to the key fundamental requirements as “must haves” and optional requirements that represented additional functionality as “could haves”, opting to only implement the non-key functionality only once the core functionality had been successfully implemented. (should haves)

* Team discussions

We made a point of having regular meetings as a team, this usually meant meeting between 1-3 times per week. During these meetings we would discuss whatever topic was held to be pertinent. As the design and development of the project progressed our meetings also incorporated discussion regarding the individual aspects of the project that we were responsible for, talking about the progress any issues that we had encountered. we also take this time to organise the groups sprints where appropriate, this was a less frequent occurrence however due to the length of the sprints themselves. ( we would then organise the sprint for the week)

* Delegated team-roles

Before we began to develop the initial user requirements for a program we decided to delegate roles within the project to the various members of the group. The roles themselves were based upon the group member’s (perceived) strengths with regard to development, all role assignments were issued with a group consensus.

We believed this to be the most efficient method to develop the application as it allowed for group members to focus their abilities in areas in which the possessed the greatest degree of competency. Additionally, following the initial planning stages, it also allowed for simultaneous development of the project.

**Design stage**

* Investigated and analysed similar systems used by competing businesses
* Began to design user-interfaces for programs (very vague not explaining what we did in detail just a vague overview of what we did)

Our first step in creating a user interface design was to examine the user interfaces of other similar systems, with particular attention paid to those utilised by competing businesses( such as Papa Johns, Pizza Hut and dominoes. We made notes regarding the aspects that we thought were good and bad and attempted to incorporate the good elements and aspects into our own design. One example is that we noticed that the majority of our competitors were limiting the amount of toppings for each pizza We also made sure to apply the principles of good UI design during the entire process.

We then began drawing concept sketches of the user interfaces for the various aspects of the project, after which we began to develop what we considered to be the more complicated applications(the mobile & web applications) into paper prototypes to properly explore their ideal full functionality as well as their usability.

* Created paper prototype of the program

We then took the

* Observed usability-test with paper-prototype to generate feedback
* Responded to feedback and made any necessary changes

**Created prototype**

* Initial user test using task-sheets to generate user feedback on the program
* Analysed feedback and re-assessed user-requirements
* Made changes to the prototype in response to feedback
* Second live usability test
* Analysed feedback and made changes

**Full development**

* Began to develop the program in full
* Challenges
* Necessary changes

**Project management**

* Dividing up the aspects of the project and assigning roles based on group members aptitudes
* Communication
  + Communication methods
  + frequency
  + Regular group meetings
* Identify and evaluate any difficulties that arose
* Project closure meeting
  + Project review
  + Lessons learnt
* Compare initial plans and what had to change due to issues
* Review risk analysis and comment
  + Did we identify the appropriate risks?
  + Were our contingency plans appropriate?

**Database Design**

* Include database diagram generated by oracle SQL diagram
* Include appropriate descriptions of your database objects

describe the database objects in great detail individually

* Include SQL CREATE statements

CREATE TABLE "ADDRESSES"   
 ( "ADDRESS\_ID" NUMBER(10,0) NOORDER NOCYCLE NOT NULL ENABLE,   
 "ADDRESS\_LINE\_1" VARCHAR2(30) NOT NULL ENABLE,   
 "ADDRESS\_LINE\_2" VARCHAR2(30) NOT NULL ENABLE,   
 "TOWN" VARCHAR2(20) NOT NULL ENABLE,   
 "POSTCODE" VARCHAR2(7) NOT NULL ENABLE,   
 CONSTRAINT "ADDRESS\_PK1" PRIMARY KEY ("ADDRESS\_ID")  
 USING INDEX ENABLE,   
 CONSTRAINT "POSTCODE\_CHK" CHECK (REGEXP\_LIKE(POSTCODE,'[A-Z]{1,2}\d{1,2}[A-Z]? \d{1}[ABD-HJLN-UW-Z]{2}')) DISABLE  
 )  
/

CREATE TABLE "ADMIN"   
 ( "ADMIN\_ID" NUMBER(10,0) GENERATED ALWAYS AS IDENTITY MINVALUE 1 MAXVALUE 9999999999999999999999999999 INCREMENT BY 1 START WITH 1 CACHE 20 NOORDER NOCYCLE NOT NULL ENABLE,   
 "BRANCH\_ID" NUMBER(10,0) NOT NULL ENABLE,   
 "FORENAME" VARCHAR2(50) NOT NULL ENABLE,   
 "SURNAME" VARCHAR2(50) NOT NULL ENABLE,   
 "USERNAME" VARCHAR2(50) NOT NULL ENABLE,   
 "PASSWORD" VARCHAR2(50) NOT NULL ENABLE,   
 CONSTRAINT "ADMIN\_PK1" PRIMARY KEY ("ADMIN\_ID")  
 USING INDEX ENABLE  
 )  
/  
ALTER TABLE "ADMIN" ADD CONSTRAINT "ADMIN\_FK" FOREIGN KEY ("BRANCH\_ID")  
 REFERENCES "BRANCH" ("BRANCH\_ID") ENABLE  
/

CREATE TABLE "BRANCH"   
 ( "BRANCH\_ID" NUMBER(10,0) NOORDER NOCYCLE NOT NULL ENABLE,   
 "BRANCH\_NAME" VARCHAR2(30) NOT NULL ENABLE,   
 "ADDRESS" VARCHAR2(50) NOT NULL ENABLE,   
 "POSTCODE" VARCHAR2(7) NOT NULL ENABLE,   
 CONSTRAINT "BRANCH\_PK" PRIMARY KEY ("BRANCH\_ID")  
 USING INDEX ENABLE  
 )  
/

CREATE TABLE "CUSTOMER"   
 ( "CUSTOMER\_ID" NUMBER(10,0) NOORDER NOCYCLE NOT NULL ENABLE,   
 "CUSTOMER\_TITLE" VARCHAR2(5) NOT NULL ENABLE,   
 "CUSTOMER\_FORENAME" VARCHAR2(20) NOT NULL ENABLE,   
 "CUSTOMER\_SURNAME" VARCHAR2(30) NOT NULL ENABLE,   
 "CUSTOMER\_EMAIL" VARCHAR2(50) NOT NULL ENABLE,   
 "CUSTOMER\_PHONE" VARCHAR2(11) NOT NULL ENABLE,   
 "CUSTOMER\_PASSWORD" VARCHAR2(256) NOT NULL ENABLE,   
 "CUSTOMER\_DOB" DATE NOT NULL ENABLE,   
 "ADDRESS\_ID" NUMBER(10,0) NOT NULL ENABLE,   
 "PASSWORD\_SALT" VARCHAR2(64),   
 CONSTRAINT "CUSTOMER\_PK" PRIMARY KEY ("CUSTOMER\_ID")  
 USING INDEX ENABLE,   
 CONSTRAINT "PASSWORD\_LENGTH\_CHK2" CHECK (LENGTHB(CUSTOMER\_PASSWORD) >= 4) ENABLE,   
 CONSTRAINT "EMAIL\_FORMAT\_CHK" CHECK (REGEXP\_LIKE(CUSTOMER\_EMAIL, '^[A-Za-z0-9.\_%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,4}$')) ENABLE,   
 CONSTRAINT "PHONE\_FORMAT\_CHK" CHECK (REGEXP\_LIKE(CUSTOMER\_PHONE,'^[0][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]$')) ENABLE  
 )  
/  
ALTER TABLE "CUSTOMER" ADD CONSTRAINT "ADDRESS\_FK" FOREIGN KEY ("ADDRESS\_ID")  
 REFERENCES "ADDRESSES" ("ADDRESS\_ID") ENABLE  
/

CREATE OR REPLACE EDITIONABLE TRIGGER "BI\_CUSTOMER"   
 before insert on "CUSTOMER"   
 for each row   
begin   
 if :NEW."CUSTOMER\_ID" is null then   
 select "CUSTOMER\_SEQ".nextval into :NEW."CUSTOMER\_ID" from sys.dual;   
 end if;   
end;   
  
/  
ALTER TRIGGER "BI\_CUSTOMER" ENABLE  
/

CREATE TABLE "DEAL"   
 ( "DEAL\_ID" NUMBER(10,0) NOCYCLE NOT NULL ENABLE,   
 "DEAL\_NAME" VARCHAR2(50) NOT NULL ENABLE,   
 "DEAL\_PRICE" NUMBER(4,2) NOT NULL ENABLE,   
 CONSTRAINT "DEAL\_PK" PRIMARY KEY ("DEAL\_ID")  
 USING INDEX ENABLE  
 )  
/

CREATE TABLE "DEAL\_BRANCH"   
 ( "DEAL\_ID" NUMBER(10,0) NOT NULL ENABLE,   
 "BRANCH\_ID" NUMBER(10,0) NOT NULL ENABLE,   
 "DEAL\_START" DATE NOT NULL ENABLE,   
 "DEAL\_END" DATE,   
 CONSTRAINT "DEAL\_BRANCH\_PK1" PRIMARY KEY ("DEAL\_ID", "BRANCH\_ID")  
 USING INDEX ENABLE  
 )  
/  
ALTER TABLE "DEAL\_BRANCH" ADD CONSTRAINT "DEAL\_BRANCH\_FK" FOREIGN KEY ("DEAL\_ID")  
 REFERENCES "DEAL" ("DEAL\_ID") ENABLE  
/  
ALTER TABLE "DEAL\_BRANCH" ADD CONSTRAINT "DEAL\_BRANCH\_FK1" FOREIGN KEY ("BRANCH\_ID")  
 REFERENCES "BRANCH" ("BRANCH\_ID") ENABLE  
/

CREATE TABLE "DEAL\_ITEM"   
 ( "DEAL\_ID" NUMBER(10,0) NOT NULL ENABLE,   
 "ITEM\_ID" NUMBER(10,0) NOT NULL ENABLE,   
 "ITEM\_QUANTITY" NUMBER(5,0) NOT NULL ENABLE,   
 CONSTRAINT "DEAL\_ITEM\_PK" PRIMARY KEY ("DEAL\_ID", "ITEM\_ID")  
 USING INDEX ENABLE  
 )  
/  
ALTER TABLE "DEAL\_ITEM" ADD CONSTRAINT "DEAL\_ITEM\_FK" FOREIGN KEY ("DEAL\_ID")  
 REFERENCES "DEAL" ("DEAL\_ID") ENABLE  
/  
ALTER TABLE "DEAL\_ITEM" ADD CONSTRAINT "DEAL\_ITEM\_FK2" FOREIGN KEY ("ITEM\_ID")  
 REFERENCES "ITEM" ("ITEM\_ID") ENABLE  
/

CREATE TABLE "DELIVERY"   
 ( "DELIVERY\_ID" NUMBER(10,0) NOORDER NOCYCLE NOT NULL ENABLE,   
 "ORDER\_ID" NUMBER(10,0) NOT NULL ENABLE,   
 "DATE\_DELIVERED" DATE NOT NULL ENABLE,   
 "DELIVERY\_NOTES" VARCHAR2(100) NOT NULL ENABLE,   
 "RIDER\_ID" NUMBER(10,0) NOT NULL ENABLE,   
 CONSTRAINT "DELIVERY\_PK" PRIMARY KEY ("DELIVERY\_ID")  
 USING INDEX ENABLE  
 )  
/  
ALTER TABLE "DELIVERY" ADD CONSTRAINT "DELIVERY\_FK" FOREIGN KEY ("ORDER\_ID")  
 REFERENCES "ORDERS" ("ORDER\_ID") ENABLE  
/  
ALTER TABLE "DELIVERY" ADD CONSTRAINT "DELIVERY\_RIDER\_FK" FOREIGN KEY ("RIDER\_ID")  
 REFERENCES "DELIVERY\_RIDER" ("RIDER\_ID") ENABLE  
/

CREATE TABLE "DELIVERY\_RIDER"   
 ( "RIDER\_ID" NUMBER(10,0) NOORDER NOCYCLE NOT NULL ENABLE,   
 "RIDER\_FORENAME" VARCHAR2(20) NOT NULL ENABLE,   
 "RIDER\_SURNAME" VARCHAR2(20) NOT NULL ENABLE,   
 "RIDER\_EMAIL" VARCHAR2(50) NOT NULL ENABLE,   
 "RIDER\_PHONE" VARCHAR2(11) NOT NULL ENABLE,   
 "RIDER\_PASSWORD" VARCHAR2(256) NOT NULL ENABLE,   
 "VEHICLE\_TYPE" VARCHAR2(20) NOT NULL ENABLE,   
 "RIDER\_DOB" DATE NOT NULL ENABLE,   
 "RIDER\_TITLE" VARCHAR2(5) NOT NULL ENABLE,   
 "PASSWORD\_SALT" VARCHAR2(64),   
 CONSTRAINT "DELIVERY\_RIDER\_PK" PRIMARY KEY ("RIDER\_ID")  
 USING INDEX ENABLE,   
 CONSTRAINT "PASSWORD\_LENGTH\_CHK3" CHECK (LENGTHB(RIDER\_PASSWORD) >= 4) ENABLE,   
 CONSTRAINT "EMAIL\_FORMAT\_CHK2" CHECK (REGEXP\_LIKE(RIDER\_EMAIL, '^[A-Za-z0-9.\_%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,4}$')) ENABLE,   
 CONSTRAINT "PHONE\_FORMAT\_CHK2" CHECK (REGEXP\_LIKE(RIDER\_PHONE,'^[0][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]$')) ENABLE  
 )  
/

CREATE TABLE "DRINK"   
 ( "DRINK\_ID" NUMBER(10,0) NOORDER NOCYCLE NOT NULL ENABLE,   
 "ITEM\_ID" NUMBER(10,0) NOT NULL ENABLE,   
 "STOCK" NUMBER(10,0) NOT NULL ENABLE,   
 "UNIT\_PRICE" NUMBER(5,5) NOT NULL ENABLE,   
 CONSTRAINT "DRINK\_PK" PRIMARY KEY ("DRINK\_ID")  
 USING INDEX ENABLE  
 )  
/  
ALTER TABLE "DRINK" ADD CONSTRAINT "DRINK\_FK" FOREIGN KEY ("ITEM\_ID")  
 REFERENCES "ITEM" ("ITEM\_ID") ENABLE  
/

CREATE TABLE "INGREDIENT"   
 ( "INGREDIENT\_ID" NUMBER(10,0) NOORDER NOCYCLE NOT NULL ENABLE,   
 "INGREDIENT\_TYPE\_ID" NUMBER(10,0) NOT NULL ENABLE,   
 "STOCK" NUMBER(10,0) NOT NULL ENABLE,   
 "UNIT\_PRICE" VARCHAR2(10) NOT NULL ENABLE,   
 "VEGETARIAN" NUMBER(1,0) NOT NULL ENABLE,   
 "VEGAN" NUMBER(1,0) NOT NULL ENABLE,   
 CONSTRAINT "INGREDIENT\_PK" PRIMARY KEY ("INGREDIENT\_ID")  
 USING INDEX ENABLE  
 )  
/  
ALTER TABLE "INGREDIENT" ADD CONSTRAINT "INGREDIENT\_FK" FOREIGN KEY ("INGREDIENT\_TYPE\_ID")  
 REFERENCES "INGREDIENT\_TYPE" ("INGREDIENT\_TYPE\_ID") ENABLE  
/

CREATE TABLE "INGREDIENT\_TYPE"   
 ( "INGREDIENT\_TYPE\_ID" NUMBER(10,0) NOORDER NOCYCLE NOT NULL ENABLE,   
 "INGREDIENT\_TYPE" VARCHAR2(50) NOT NULL ENABLE,   
 "DESCRIPTION" VARCHAR2(100) NOT NULL ENABLE,   
 CONSTRAINT "INGREDIENT\_TYPE\_PK" PRIMARY KEY ("INGREDIENT\_TYPE\_ID")  
 USING INDEX ENABLE  
 )  
/

CREATE TABLE "ITEM"   
 ( "ITEM\_ID" NUMBER(10,0) NOORDER NOCYCLE NOT NULL ENABLE,   
 "ITEM\_NAME" VARCHAR2(50) NOT NULL ENABLE,   
 "ITEM\_TYPE" VARCHAR2(30) NOT NULL ENABLE,   
 "SELLING\_PRICE" VARCHAR2(30) NOT NULL ENABLE,   
 CONSTRAINT "ITEM\_PK" PRIMARY KEY ("ITEM\_ID")  
 USING INDEX ENABLE  
 )  
/

CREATE TABLE "MENU"   
 ( "MENU\_ID" NUMBER(10,0) NOORDER NOCYCLE NOT NULL ENABLE,   
 "BRANCH\_ID" NUMBER(10,0) NOT NULL ENABLE,   
 "PRICE\_MULTIPLIER" NUMBER(5,5) NOT NULL ENABLE,   
 CONSTRAINT "MENU\_PK" PRIMARY KEY ("MENU\_ID")  
 USING INDEX ENABLE  
 )  
/  
ALTER TABLE "MENU" ADD CONSTRAINT "MENU\_FK" FOREIGN KEY ("BRANCH\_ID")  
 REFERENCES "BRANCH" ("BRANCH\_ID") ENABLE  
/

CREATE TABLE "MENU\_ITEM"   
 ( "MENU\_ID" NUMBER(10,0) NOT NULL ENABLE,   
 "ITEM\_ID" NUMBER(10,0) NOT NULL ENABLE,   
 "QUANTITY" NUMBER(5,0) NOT NULL ENABLE,   
 "UNIT\_PRICE" NUMBER(5,5) NOT NULL ENABLE,   
 CONSTRAINT "MENU\_ITEM\_PK" PRIMARY KEY ("MENU\_ID", "ITEM\_ID")  
 USING INDEX ENABLE  
 )  
/  
ALTER TABLE "MENU\_ITEM" ADD CONSTRAINT "MENU\_ITEM\_FK" FOREIGN KEY ("MENU\_ID")  
 REFERENCES "MENU" ("MENU\_ID") ENABLE  
/  
ALTER TABLE "MENU\_ITEM" ADD CONSTRAINT "MENU\_ITEM\_FK2" FOREIGN KEY ("ITEM\_ID")  
 REFERENCES "ITEM" ("ITEM\_ID") ENABLE  
/

CREATE TABLE "ORDER\_ITEM"   
 ( "ITEM\_ID" NUMBER(10,0) NOT NULL ENABLE,   
 "ORDER\_ID" NUMBER(10,0) NOT NULL ENABLE,   
 "QUANTITY" NUMBER(10,0) NOT NULL ENABLE,   
 "UNIT\_PRICE" VARCHAR2(20) NOT NULL ENABLE,   
 CONSTRAINT "ORDER\_ITEM\_PK" PRIMARY KEY ("ITEM\_ID", "ORDER\_ID")  
 USING INDEX ENABLE  
 )  
/  
ALTER TABLE "ORDER\_ITEM" ADD CONSTRAINT "ORDER\_ITEM\_FK" FOREIGN KEY ("ITEM\_ID")  
 REFERENCES "ITEM" ("ITEM\_ID") ENABLE  
/  
ALTER TABLE "ORDER\_ITEM" ADD CONSTRAINT "ORDER\_ITEM\_FK2" FOREIGN KEY ("ORDER\_ID")  
 REFERENCES "ORDERS" ("ORDER\_ID") ENABLE  
/

CREATE TABLE "PIZZA"   
 ( "PIZZA\_ID" NUMBER(10,0) NOORDER NOCYCLE NOT NULL ENABLE,   
 "ITEM\_ID" NUMBER(10,0) NOT NULL ENABLE,   
 "UNIT\_PRICE" VARCHAR2(10) NOT NULL ENABLE,   
 CONSTRAINT "PIZZA\_PK" PRIMARY KEY ("PIZZA\_ID")  
 USING INDEX ENABLE  
 )  
/  
ALTER TABLE "PIZZA" ADD CONSTRAINT "PIZZA\_FK" FOREIGN KEY ("ITEM\_ID")  
 REFERENCES "ITEM" ("ITEM\_ID") ENABLE  
/

CREATE TABLE "PIZZA\_INGREDIENT"   
 ( "PIZZA\_ID" NUMBER(10,0) NOT NULL ENABLE,   
 "INGREDIENT\_ID" NUMBER(10,0) NOT NULL ENABLE,   
 "QUANTITY\_USED" NUMBER(10,0) NOT NULL ENABLE,   
 CONSTRAINT "PIZZA\_INGREDIENT\_PK1" PRIMARY KEY ("PIZZA\_ID", "INGREDIENT\_ID")  
 USING INDEX ENABLE  
 )  
/  
ALTER TABLE "PIZZA\_INGREDIENT" ADD CONSTRAINT "PIZZA\_INGREDIENT\_FK" FOREIGN KEY ("PIZZA\_ID")  
 REFERENCES "PIZZA" ("PIZZA\_ID") ENABLE  
/  
ALTER TABLE "PIZZA\_INGREDIENT" ADD CONSTRAINT "PIZZA\_INGREDIENT\_FK2" FOREIGN KEY ("INGREDIENT\_ID")  
 REFERENCES "INGREDIENT" ("INGREDIENT\_ID") ENABLE  
/

CREATE TABLE "RIDER\_ACTIVITY"   
 ( "SHIFT\_ID" NUMBER(10,0) NOORDER NOCYCLE NOT NULL ENABLE,   
 "RIDER\_ID" NUMBER(10,0) NOT NULL ENABLE,   
 "SHIFT\_START" DATE,   
 "SHIFT\_END" DATE,   
 "SHIFT\_LENGTH" VARCHAR2(20),   
 "LOCATION" VARCHAR2(100),   
 "DELIVERIES\_MADE" NUMBER(4,0),   
 "STATUS" VARCHAR2(100),   
 "TOTAL\_EARNED" VARCHAR2(10) NOT NULL ENABLE,   
 CONSTRAINT "RIDER\_ACTIVITY\_PK" PRIMARY KEY ("SHIFT\_ID")  
 USING INDEX ENABLE  
 )  
/  
ALTER TABLE "RIDER\_ACTIVITY" ADD CONSTRAINT "RIDER\_ACTIVITY\_FK" FOREIGN KEY ("RIDER\_ID")  
 REFERENCES "DELIVERY\_RIDER" ("RIDER\_ID") ENABLE  
/

CREATE TABLE "SIDE"   
 ( "SIDE\_ID" NUMBER(10,0) NOCYCLE NOT NULL ENABLE,   
 "ITEM\_ID" NUMBER(10,0) NOT NULL ENABLE,   
 "STOCK" NUMBER(10,0) NOT NULL ENABLE,   
 "UNIT\_PRICE" NUMBER(5,5) NOT NULL ENABLE,   
 CONSTRAINT "SIDE\_PK" PRIMARY KEY ("SIDE\_ID")  
 USING INDEX ENABLE  
 )  
/  
ALTER TABLE "SIDE" ADD CONSTRAINT "SIDE\_FK" FOREIGN KEY ("ITEM\_ID")  
 REFERENCES "ITEM" ("ITEM\_ID") ENABLE  
/

CREATE TABLE "TRANSACTION"   
 ( "TRANSACTION\_ID" NUMBER(10,0)NOORDER NOCYCLE NOT NULL ENABLE,   
 "ORDER\_ID" NUMBER(10,0) NOT NULL ENABLE,   
 "PAYMENT\_METHOD" VARCHAR2(20) NOT NULL ENABLE,   
 "DETAIL\_1" VARCHAR2(20),   
 "DETAIL\_2" VARCHAR2(20),   
 "DETAIL\_3" VARCHAR2(20),   
 "DETAIL\_4" VARCHAR2(20),   
 "TRANSACTION\_VALUE" NUMBER(10,3) NOT NULL ENABLE,   
 CONSTRAINT "TRANSACTION\_PK" PRIMARY KEY ("TRANSACTION\_ID")  
 USING INDEX ENABLE  
 )  
/  
ALTER TABLE "TRANSACTION" ADD CONSTRAINT "TRANSACTION\_FK" FOREIGN KEY ("ORDER\_ID")  
 REFERENCES "ORDERS" ("ORDER\_ID") ENABLE  
/

* Submit normalised tables and compare your schema with you class diagram

Screenshot of comparison of old erd and finished flused

**Requirements analysis and design**

* list functional and non-functional requirements
* analysis and design
  + use case diagram(with some basic descriptions) for the final version
  + user stories with acceptance tests consistent with the use cases identified in the use case model
  + user requirements list(product backlog)
  + design level sequence diagram
  + Evaluation on how the design changes during implementation.

**Conclusion**

In conclusion there were a whole multitude of features that we would of liked to of had which we didn’t add due to time constraints.