F17/1742/2019

KYUNGU SAMUEL MUTINDA

BACHELOR OF SCIENCE IN ELECTRICAL AND ELECTRONIC ENGINEERING.

FEE232 DBMS PROJECT DOCUMENTATION

AtoPC ONLINE ELECTRONICS STORE

ABSTRACT:

This is a documentation of the FEE232 project on database management systems that was given as part of the coursework for the unit. The documentation is intended to explain the design process that took place during the conception of the project, list the tools used to implement the actual project, explain the functionality of the project and the problems encountered during conception.

For the sake of order, the documentation has been divided into several sections, each touching on a specific area of the project.

The first section of the documentation is a basic introduction to the project, that is: the problem that the students were meant to solve and also an introduction to the various tools used to implement the project. Reference can be made to the project files that were submitted alongside this documentation.

The second section of the documentation lists previous solutions to problems related to the one at hand, goes over the implementation of those solutions and lists the tools used in implementation. Comparisons have been made with my implementations and the inspiration drawn from the previous projects has also been listed.

The third section of the documentation dissects the actual project and explains the design process and the implementation of functionality in high detail. Visual aids such as diagrams have been included together with code snippets to demystify the implementation of the project.

The fourth section explains some of the difficulties faced, the ones that were overcome or worked around and also the ones that could not be overcome. All the workarounds and the solutions have also been explained.

The fifth section discusses possible refinements to the project at its current state if more resources (including time) were available.

The final section explains whether or not the initially stated objectives were achieved and to what extent.

SECTION1: INTRODUCTION:

The project under discussion intended to enforce the concepts of relational database management systems by putting them into practice, specifically through the implementation of an web application, specifically an online electronics store.

All the programming was done on Visual studio code and DB browser for sqlite. The languages used are; html, css, SQL and python. The uses for each language have been explained later in the introduction.

The web app consists of two sections, the front end and the back end, the back end being the most important since it put these concepts of database management systems into practice. For this particular problem, a relational database management system was required to manage inventory, delivery information, payment information and customer information. The back end framework used for the web application is Flask and the database engine of choice is SQLite. SQLite allowed storage, manipulation and querying of the data while flask allowed integration of the web app with the database. SQLalchemy, which is part of flask was also used for queries.

The front end of the project was implemented using html5 and CSS. Rendering of static data was done primarily using html while all the styling was done using css. Html brought up some bottlenecks when rendering dynamic data. I therefore had to make use of Jinja together with my html to render such data.

These tools were chosen on the basis of ease of use and streamlined workflow. They were however insufficient especially in rendering of dynamic data, a task which languages such as Javascript and PHP would have done an excellent job.

The problem given was building a web app for an electronics enterprise vendor from which customers could purchase electronics and inventory could be managed. My implementation is a web app for a mock enterprise called “AtoPc” which deals in the sale of consumer electronics such as smartphones, laptops, earphones and headphones. The web app has a customer section and an admin section. Customers can access the web app from their browser, shop for electronics, keep track of their orders and previous purchases. Inventory clerks can also access the web app and manage inventory by adding new inventory and keeping track of available inventory. The marketing team can access sales data from the admin section and customer care can also keep track of orders in case of any issues.

Purchase of devices is done by configuring the specification of the device. For example, if a customer needs to buy a smartphone, they can select the model and configure the storage and color of the smartphone. The device is then delivered to the user’s post office of choice or the customer’s location based on the customer’s preference. A cash payment

or a contract payment is allowed. The customer can view information about their purchase and delivery from their dashboard.

SECTION 2: PREVIOUS IMPLEMENTATIONS:

ideas and inspiration from the database were partly borrowed from other implementation of the online electronics stores such as Samsung's own web store (www.samsung.com), ebay (www.ebay.com), apple's web store (www.apple.com).

The three use html and css for front end and proprietary web frameworks together with Javascript and PHP. Data is managed by amazon web services engine. The framework and database technologies are very complex and mostly proprietary but front end is primarily html and css.

Samsung and apple use a configuration method for purchases where the customers can configure their devices to their liking which is very similar to my implementation.. the problem with their implementations however is that customers can only purchase devices from one manufacturer. Ebay does not involve configuration, which clutters the website and makes it hard to access a particular configuration of a device. The advantage of their implementation is that it allows customers to choose products from numerous manufacturers.

My implementation tried to hybridize these two different implementations in order to make it easy for customers to purchase products by allowing them to configure and at the same time give them the freedom to purchase from different manufacturers

SECTION 3: METHODS:

The design process involved designing the back end and the front end. Design work for the back end involved designing the logical schema while the front end design work involved drafting user interfaces.

The nature of the project is that it involves a wide variety of data all of which has to be linked in such a way that it has as few anomalies as possible. Careful problem analysis and design was paramount. I set out to satisfy the following requirements in my database design:

1. The database should hold information about the customers including login information such as a unique and valid email address and username, and encrypted passwords for login.
2. The database should hold information about the inventory, categorized according to the type of product eg: laptops, smartphones etc.
3. The database should hold information about orders, specifically the details of the purchased device, the customer who ordered the device, the tracking number of the package, the date of purchase, the date of delivery, the company in charge of delivery, the location of delivery and the delivery address.
4. The database should hold information about all payments made by customers, whether in form of contract payments or instant payments. Information to be stored includes : the credit card number used in making the payment, the date of payment, the amount paid and the customer who makes the payment.
5. The database should hold information about ongoing contracts, the starting and ending dates and the monthly deductions.

With these requirements in consideration, I was able to begin the design process. The logical schema was drafted in the form of an entity relationship diagram which has been illustrated in the next page. It serves as a concise summary of the database design. All entities, relationships and constraints have been indicated on the diagram. The ER diagram was then used to design the relational schema which has also been indicated in the later pages.

The schema went through several stages of fine tuning as project implementation proceeded for the sake of normalization and satisfaction of newly discovered requirements during implementation. After drafting, the tables were created in sqlite using SQL statements which have also been given in the later parts of this section. In order to make use of the sqlalchemy syntax provided in flask for python – like queries, I created classes

for the tables in my modules python file. The code has been attached in the appendix section of the documentation.

Page 7 contains the entity relationship diagram, pages 8 to 11 contain the entities and page 9 contains the sql statements for each table

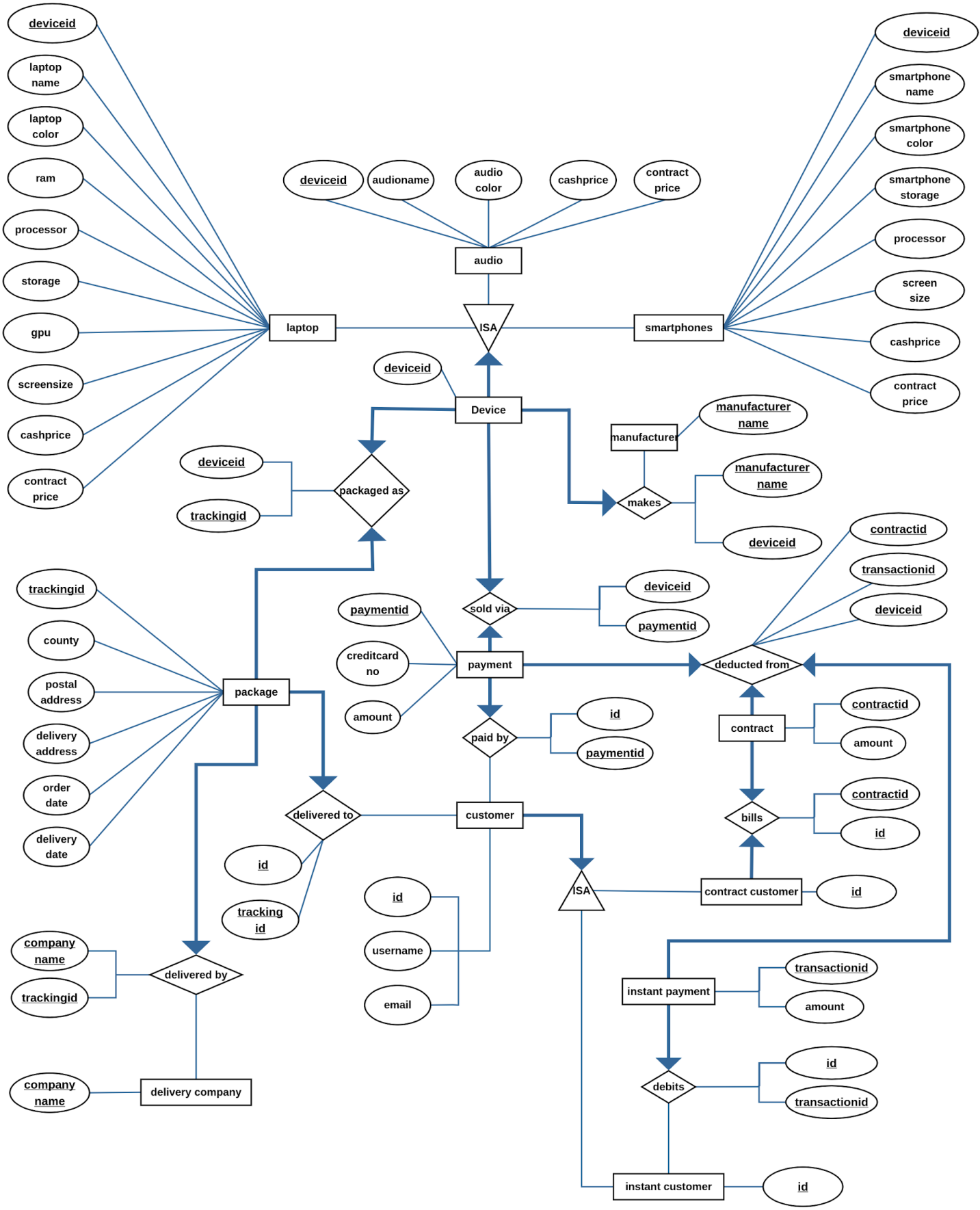
The next step of the design process was to design the interface. I settled on a web page design that contains 3 sections: a navigation bar, a body and a footer. The implementation of the three sections was different for the customer side and for the admin side since general navigation was different for each.

For the customer side, the customer can navigate to the different product category pages, that is: smartphones, laptops and audio. They can also navigate to the homepage if they so wish. If the customer is logged out he or she has an option to navigate to the login page by clicking on the login button or navigate to the register page by clicking on the register button in case he or she has no account. There is also an extra admin login/register button that allows the employees to access the admin side of the website. In order to prevent regular users from registering as administrator, each employee who is not registered has to enter a security key which allows them to create administrator accounts. For the admin side, the navigation bar allows the admin user to check the dashboard, manage stock for audio devices, laptops and smartphones. The users can also log out using the logout button on the navigation bar.

The body of the website contains all the data that users need to see. It of course is different for each webpage. The pages have been built with ease of navigation as a primary consideration. The homepage gives a brief overview of the products available on the site. Users can purchase products on the home page directly or navigate to the category of the product they need to purchase from the navigation bar. Each device category page consists of top picks (devices of that category which are recommended to the user) and brand logos which users can click on and access models of devices from that manufacturer. From there on, the customer can select the model of their choice, which takes them to the configuration screen. The configuration screen allows a user to pick the device configuration they intend to purchase. After submitting the configuration, the user can proceed to enter delivery and payment information.

A dashboard page is provided for both customers and admins. It gives information at a glance. For the customer, it shows them information about purchases, orders and payments while for admins, it shows information about sales and deliveries.

Admins can restock devices from the manage devices section (more of this explained in the demo video). They can configure devices to stock and specify the amount of units they wish to restock.


deviceid
laptop name
laptop color
ram
processor
storage
gpu
screensize
cashprice
contract price
laptop
deviceid
audioname
audio color
cashprice
contract price
audio
ISA
smartphones
deviceid
smartphone name
smartphone color
smartphone storage
processor
screen size
cashprice
contract price
deviceid
Device
manufacturer
manufacturer name
deviceid
trackingid
packaged as
makes
manufacturer name
deviceid
trackingid
county
postal address
delivery address
order date
delivery date
package
paymentid
creditcard no
amount
sold via
payment
deviceid
paymentid
deducted from
contractid
transactionid
deviceid
id
paymentid
paid by
contract
contractid
amount
delivered to
customer
bills
contractid
id
id
tracking id
ISA
contract customer
id
id
username
email
company name
trackingid
delivered by
delivery company
company name
instant payment
transactionid
amount
debits
id
transactionid
instant customer
id

Tables:

Audio

	deviceid (FK)	audioname	audiocolor	contract_price	cash_price
	Filter	Filter	Filter	Filter	Filter
1	1	AirPods	white	2500	15000

Bills

	customerid (FK)	contractid (FK)
	Filter	Filter
1	1	3

Contract

	contractid	:ontractamoun	startingdate	endingdate
	Filter	Filter	Filter	Filter
1	4	1250	2021-06-03	2022-06-03

Contractcustomer

	id (FK)
	Filter
1	2
2	1

Customer

	id	username	email	password_hash
	Filter	Filter	Filter	Filter
1	1	Samuel ...	mutindasa...	$2b$12$TZ...

Debits

	customerid (FK)	transactionid (FK)
	Filter	Filter
1	1	1

Deducted_from

	paymentid (FK)	contractid (FK)	transactionid (FK)
	Filter	Filter	Filter
1	1	*NULL*	2
2	2	4	*NULL*

Delivered_by

	trackingid (FK)	companyname (FK)
	Filter	Filter
1	1	DHL

Delivered_to

	trackingid (FK)	customerid (FK)
	Filter	Filter
1	1	3

Delivery_company

	companyname
	Filter
1	DHL
2	Posta

Device

	deviceid
	Filter
1	1
2	2

Instantpayment

	transactionid	transactionamount	transactiondate
	Filter	Filter	Filter
1	3	30000	2021-06-03
2	4	20000	2021-06-04

Laptops

	deviceid (FK)	laptopname	laptopcolor	ram	processor	storage	gpu	screensize	cash_price	contract_price
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1	MacBook Air(M1)	space gray	8gb	Apple M1	256gb	Apple M1 graphics	13 inch	10834	130000
2	2	MacBook Air(M1)	space gray	8gb	Apple M1	256gb	Apple M1 graphics	13 inch	10834	130000

Makes

	manufacturername (FK)	deviceid (FK)
	Filter	Filter
1	Apple	1

Manufacturer

	manufacturername
	Filter
1	Apple
2	Samsung

Package

	trackingid	county	delivery_address	postal_address	delivery_date	order_date
	Filter	Filter	Filter	Filter	Filter	Filter
1	1	nairobi	2100 nairobi	21333	2021-06-04	2021-06-03
2	2	nairobi	2100 nairobi	21333	2021-06-04	2021-06-03

Packaged_as

	deviceid (FK)	trackingid (FK)
	Filter	Filter
1	2	2
2	1	1

Paid_by

	paymentid (FK)	customerid (FK)
	Filter	Filter
1	1	1
2	2	2

Payment

	paymentid	paymentamount	credit_card
	Filter	Filter	Filter
1	6	1250	123456
2	7	30000	123456

Instantcustomer

	id (FK)
	Filter
1	2
2	1

Smartphones

	deviceid (FK)	smartphonename	smartphonecolor	smartphonestorage	processor	screensize	cash_price	contract_price
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	4	iPhone XR	coral	64gb	Apple A12	6.1 inch	60000	5000
2	5	iPhone XR	red	64gb	Apple A12	6.1 inch	60000	5000

Soldvia

	deviceid (FK)	paymentid (FK)
	Filter	Filter
1	4	3
2	2	2
3	1	1

SQL STATEMENTS:

```
CREATE TABLE "audio" (
        "deviceid"      INTEGER NOT NULL,
        "audioname" TEXT NOT NULL,
        "audiocolor"   TEXT NOT NULL,
        "contract_price"        NUMERIC NOT NULL,
        "cash_price"  NUMERIC NOT NULL,
        FOREIGN KEY("deviceid") REFERENCES "device"("deviceid") ON DELETE
        CASCADE,
        PRIMARY KEY("deviceid")
);

CREATE TABLE "bills" (
        "customerid"  INTEGER NOT NULL,
        "contractid"    INTEGER NOT NULL,
        FOREIGN KEY("contractid") REFERENCES "contract"("contractid") ON DELETE
        CASCADE,
        FOREIGN KEY("customerid") REFERENCES "contractcustomer"("id") ON DELETE
        CASCADE,
        PRIMARY KEY("customerid","contractid")
);

CREATE TABLE "contract" (
        "contractid"    INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
        "contractamount"      NUMERIC NOT NULL,
        "startingdate"INTEGER NOT NULL,
        "endingdate"  INTEGER NOT NULL
);

CREATE TABLE contractcustomer (
        id INTEGER NOT NULL,
        PRIMARY KEY (id),
```

```
        FOREIGN KEY(id) REFERENCES customer (id)
);

CREATE TABLE customer (
        id INTEGER NOT NULL,
        username VARCHAR(30) NOT NULL,
        email VARCHAR(50) NOT NULL,
        password_hash VARCHAR(60) NOT NULL,
        PRIMARY KEY (id),
        UNIQUE (username),
        UNIQUE (email)
);



CREATE TABLE "debits" (
        "customerid"  INTEGER NOT NULL,
        "transactionid"         INTEGER NOT NULL,
        FOREIGN KEY("transactionid") REFERENCES "instantpayment"("transactionid")
        ON DELETE CASCADE,
        FOREIGN KEY("customerid") REFERENCES "instantcustomer"("id"),
        PRIMARY KEY("customerid","transactionid")
);

CREATE TABLE "deducted_from" (
        "paymentid"   INTEGER NOT NULL,
        "contractid"    INTEGER,
        "transactionid"         INTEGER,
        FOREIGN KEY("contractid") REFERENCES "contract"("contractid") ON DELETE
        CASCADE,
        FOREIGN KEY("transactionid") REFERENCES "instantpayment"("transactionid")
        ON DELETE CASCADE,
        PRIMARY KEY("paymentid","contractid","transactionid"),
        FOREIGN KEY("paymentid") REFERENCES "payment"("paymentid") ON DELETE
        CASCADE
);

CREATE TABLE "delivered_by" (
        "trackingid"    INTEGER NOT NULL,
        "companyname"       TEXT NOT NULL,
        FOREIGN KEY("companyname") REFERENCES
        "delivery_company"("companyname") ON DELETE CASCADE,
        FOREIGN KEY("trackingid") REFERENCES "package"("trackingid") ON DELETE
        CASCADE,
        PRIMARY KEY("trackingid","companyname")
);
```

```
CREATE TABLE "delivered_to" (
        "trackingid"    INTEGER NOT NULL,
        "customerid"  INTEGER NOT NULL,
        PRIMARY KEY("trackingid","customerid"),
        FOREIGN KEY("customerid") REFERENCES "customer"("id") ON DELETE
        CASCADE,
        FOREIGN KEY("trackingid") REFERENCES "package"("trackingid") ON DELETE
        CASCADE
);

CREATE TABLE delivery_company (
        companyname TEXT NOT NULL,
        PRIMARY KEY (companyname)
);


CREATE TABLE "device" (
        "deviceid"      INTEGER NOT NULL UNIQUE,
        PRIMARY KEY("deviceid")
);

CREATE TABLE "instantpayment" (
        "transactionid"         INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
        "transactionamount"INTEGER NOT NULL,
        "transactiondate"     DATE NOT NULL
);

CREATE TABLE "laptops" (
        "deviceid"      INTEGER NOT NULL UNIQUE,
        "laptopname"TEXT NOT NULL,
        "laptopcolor"  TEXT NOT NULL,
        "ram"  TEXT NOT NULL,
        "processor"   TEXT NOT NULL,
        "storage"       TEXT NOT NULL,
        "gpu"  TEXT NOT NULL,
        "screensize"  TEXT NOT NULL,
        "cash_price"  NUMERIC NOT NULL,
        "contract_price"       NUMERIC NOT NULL,
        PRIMARY KEY("deviceid"),
        FOREIGN KEY("deviceid") REFERENCES "device"("deviceid") ON DELETE
        CASCADE
);

CREATE TABLE "makes" (
        "manufacturername"          TEXT NOT NULL,
```

```
	"deviceid"	INTEGER NOT NULL,
	FOREIGN KEY("deviceid") REFERENCES "device"("deviceid") ON DELETE
	CASCADE,
	PRIMARY KEY("manufacturername","deviceid")
);

CREATE TABLE "manufacturer" (
	"manufacturername"		TEXT NOT NULL UNIQUE,
	PRIMARY KEY("manufacturername")
);

CREATE TABLE "package" (
	"trackingid"	INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
	"county"	TEXT NOT NULL,
	"delivery_address"	TEXT,
	"postal_address"	TEXT,
	"delivery_date"	DATE NOT NULL,
	"order_date"	DATE NOT NULL
);
CREATE TABLE "packaged_as" (
	"deviceid"	INTEGER NOT NULL,
	"trackingid"	INTEGER NOT NULL,
	FOREIGN KEY("trackingid") REFERENCES "package"("trackingid") ON DELETE
	CASCADE,
	PRIMARY KEY("deviceid","trackingid"),
	FOREIGN KEY("deviceid") REFERENCES "device"("deviceid")
);

CREATE TABLE "paidby" (
	"paymentid"	INTEGER NOT NULL,
	"customerid"	INTEGER NOT NULL,
	FOREIGN KEY("customerid") REFERENCES "customer"("id") ON DELETE
	CASCADE,
	FOREIGN KEY("paymentid") REFERENCES "payment"("paymentid") ON DELETE
	CASCADE,
	PRIMARY KEY("paymentid","customerid")
);

CREATE TABLE "payment" (
	"paymentid"	INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
	"paymentamount"	NUMERIC NOT NULL,
	"credit_card"	TEXT NOT NULL
);

CREATE TABLE "instantcustomer" (
	"id"	INTEGER NOT NULL,
```

```
        PRIMARY KEY("id"),
        FOREIGN KEY(id) REFERENCES customer(id) ON DELETE CASCADE
);

CREATE TABLE "smartphones" (
        "deviceid"      TEXT NOT NULL UNIQUE,
        "smartphonecolor"   TEXT NOT NULL,
        "smartphonename"  TEXT NOT NULL,
        "smartphonestorage"         TEXT NOT NULL,
        "processor"   TEXT NOT NULL,
        "screensize"  TEXT NOT NULL,
        "cash_price"  NUMERIC NOT NULL,
        "contract_price"       INTEGER NOT NULL,
        PRIMARY KEY("deviceid"),
        FOREIGN KEY("deviceid") REFERENCES "device"("deviceid") ON DELETE
        CASCADE
);




CREATE TABLE "soldvia" (
        "deviceid"      INTEGER NOT NULL,
        "paymentid"   INTEGER NOT NULL,
        FOREIGN KEY("paymentid") REFERENCES "payment"("paymentid") ON DELETE
        CASCADE,
        FOREIGN KEY("deviceid") REFERENCES "device"("deviceid"),
        PRIMARY KEY("deviceid","paymentid")
);
```

SECTION 4: DIFFICULTIES FACED:

Some of the difficulties faced during the implementation of this project include:

- Poor dynamic rendering of data due to lack of experience in javascript. Flask allows dynamic rendering of data using python through jinja syntax which has very limited functionality. This placed bottlenecks on some of the ideas I had which led to undesired workarounds and sometimes discarding of ideas. An example is on the configure screen where originally I had planned that the current price of the configuration be indicated live while the customer picks out the product options. This was however not possible using only html, css and jinja but could easily be fixed using Javascript.

- The problem specification was a bit vague. Appropriate details on the general outlook of the website and most of the website's functionality were missing. Some of these issues were discovered during implementation and often required a rework of the design and hence reconstruction of already implemented sections of the project which was tedious. This was not a major issue since by the end of the implementation, most of the issues were taken into consideration and fixed.

Most of the issues encountered were due to the limited capability of Jinja in rendering of dynamic data and could be solved if more time was available for the sake of learning Javascript.

SECTION 5: POSSIBLE IMPROVEMENTS:

- Better dynamic rendering of dynamic data using Javascript. This would improve the front end and the general user friendliness of the website significantly.

- More data analysis tools in the admin page such as graphs for the sake of analysis of sales data by marketing teams and sales teams in general.

- Better automation for restocking in the admin page to easen and quicken the restocking process. At the current state, the admin has to manually enter the price of the device that they wish to restock which can get very tedious when dealing with large stock.

- Tweaking of the front end to make it more mobile friendly and responsive.

- General improvements in the performance and efficiency of the website and the database. The current scheduling of queries and transactions has been implemented serially which is not appropriate for concurrent access and good database performance. The python code could also be improved efficiency-wise to improve the performance of the website.

SECTION 6: CONCLUSION:

The project was successfully completed with only a few issues all being owed to a limited amount of time. Most of the original objectives were achieved. All the design work was done properly, the implementation was carried out with very few issues and with no hanging functionality. Proper testing was also done and all issues rectified.

The website has a visually appealing, refined and easy to use front end that is well integrated with the back end with no inconsistencies. The back end has a working database that has been refined to reduce inconsistencies and anomalies. The middleware has also been implemented properly and been tested to ensure that each component of the front end has been appropriately integrated to the back end.

Most of the issues encountered were worked around while unnecessary functionality was saved for future improvements.