

Software Design Description

January 3, 2020

FDelivrey

Sarah Hananayev
Raphael Gozlan
Elchanan Mahatsri
Avichai Israeli

Table of Contents

1.0. Introduction	1
1.1. Purpose	1
1.2. Scope	1
1.3. overview	1
2.0. system overview	4
3.0. system Architecture	5
3.1 Architectural Design	5
3.1.1 business	6
3.1.2 courier	8
3.2 Decomposition Description	8
3.3 Design Rationale	
4.0. Data Structure Design	9
4.1. business Table Database	9
4.1.1. User Table	10
4.1.2. Couriers Table	10
4.1.3. deliveries Table	10
4.1.4 Payment Table	
4.2. Data Dictionary	11
4.2.1 business	11
4.2.2 courier	11
5.0. Component Design	12
5.1 business Component	
5.2 Courier Component	
6.0. User Interface Design	17
6.1 Store owner user Interface	17
6.2 Screen Images	19
7.0 Requirements Matrix	21

1.0. Introduction

1.1. Purpose

This software design document describes the architecture and system design of FDelivery. This includes the architectural features of the system .The primary audiences of this document are the software developers.

1.2. Scope

There are two parts of this system. The Store owner & courier.

The communication between two parts is standard client-server architecture with a database on the server. This system does interface with external system because of the payment in the app after the delivery

The advantage of the app is its ease of use, its availability, reliability and the large selection of couriers it offers.

The downside of the app is that it is based on a mass database and may take time to distribute the app and gather the human pool of couriers and businesses needed for an excellent app.

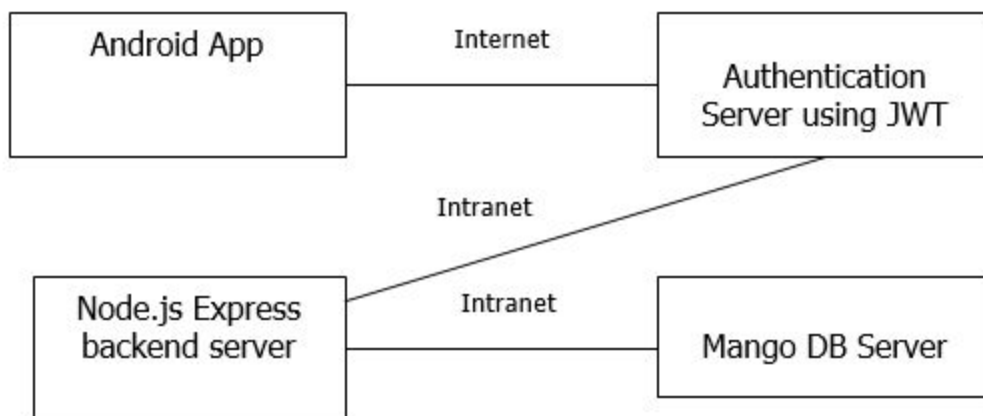
2.0. SYSTEM OVERVIEW

In general the app allows a user to register to the app as a Business or a Courier by filling in basic personal details and authenticating through FireBase back-end Authentication server.

A Business accesses the On-Demand Delivery functionality through the App. The On-Demand Delivery resides on a dedicated Node.js Express back-end Server with a permanent and real-time connection. It allows the Business to request a delivery with a costume price for the Couriers to confirm / reject the delivery which will be saved in a MongoDB database.

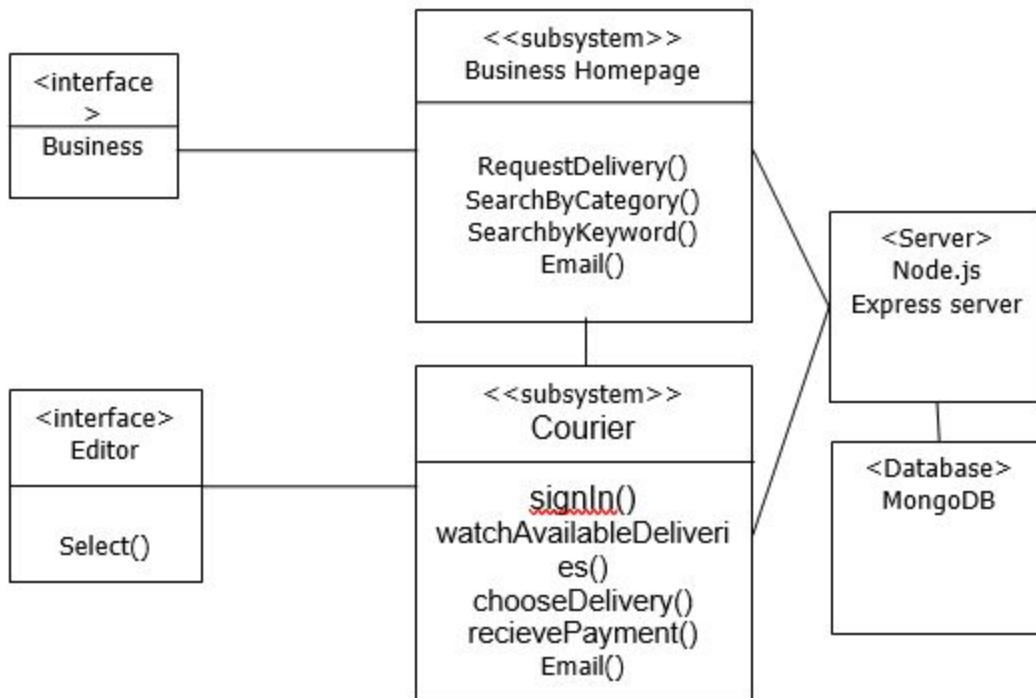
The Courier gets all of the available deliveries to be made from the Express Server and can choose from a variety of deliveries he wishes to make.

The On-demand delivery is designed for multiple users but the concurrent usage is handled client-side. That is, the app will execute on the client (user's) phone and will make requests to the On-Line server. These requests are handled sequentially by the server with no transient data storage



3. Architectural Design

3.1+3.2



Please note that this top-level design starts with the Domain Classes, which are subsystems in the design.

Business

Name: business

Type: Subsystem

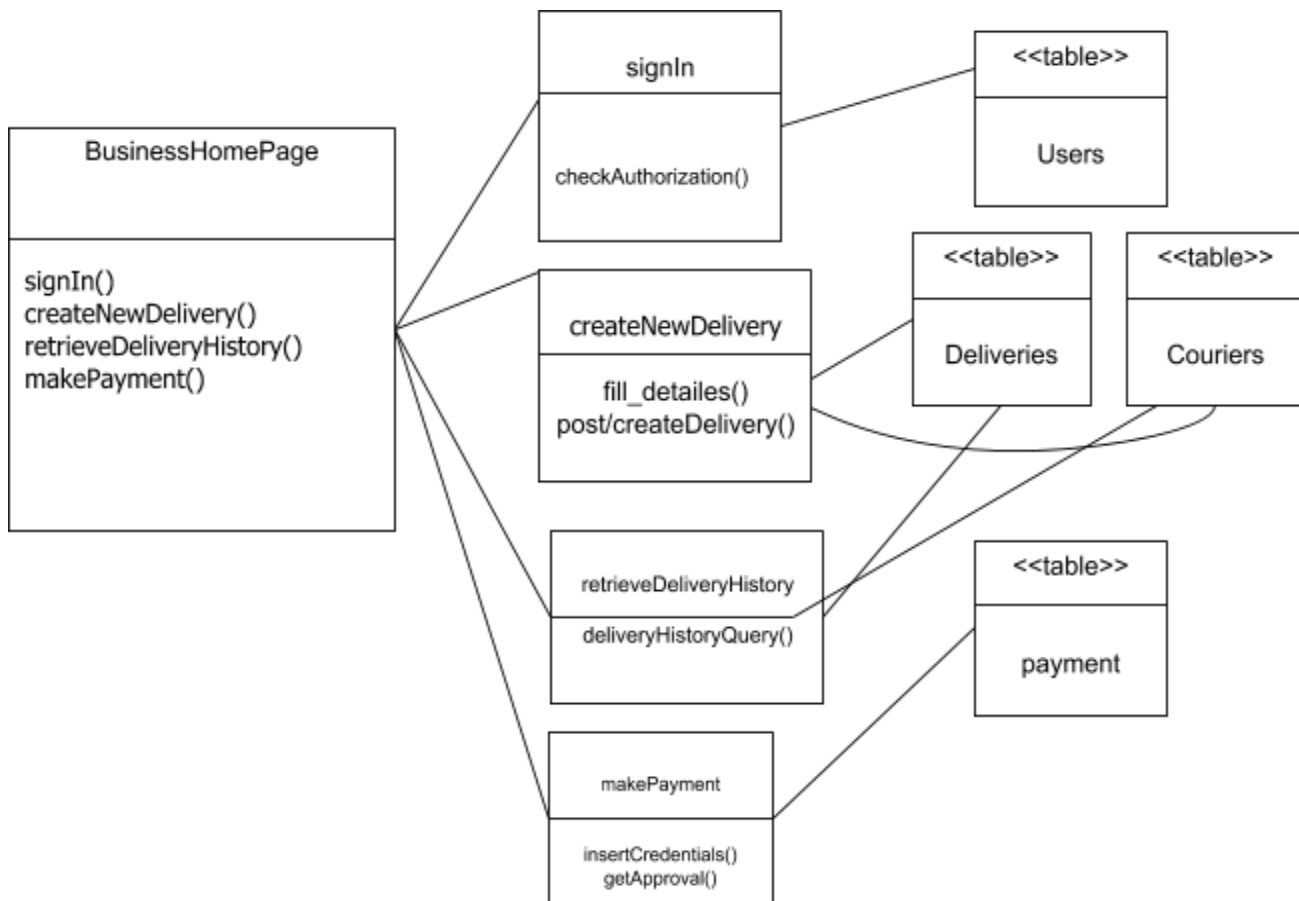
Description: This is the primary entrance to the system for business owners.

Resources: authorization, access to DB, external interface of payment.

Operations (detailed below):

- `signIn()`
- `createNewDelivery()`
- `retrieveDeliveryHistory()`
- `makePayment()`

Unit Design:



user Table-

contain a list of all the business using the app

deliveries Table-

contain a list of all the deliveries ordered in the app

couriers Table-

contain a list of all the couriers using the app

payment Table-

contain a list of all payment status for all the orders

courier

Name: courier

Type: Subsystem

Description: This is the primary entrance to the system for courier

Resources: authorization, access to DB, external interface of payment.

Operations (detailed below):

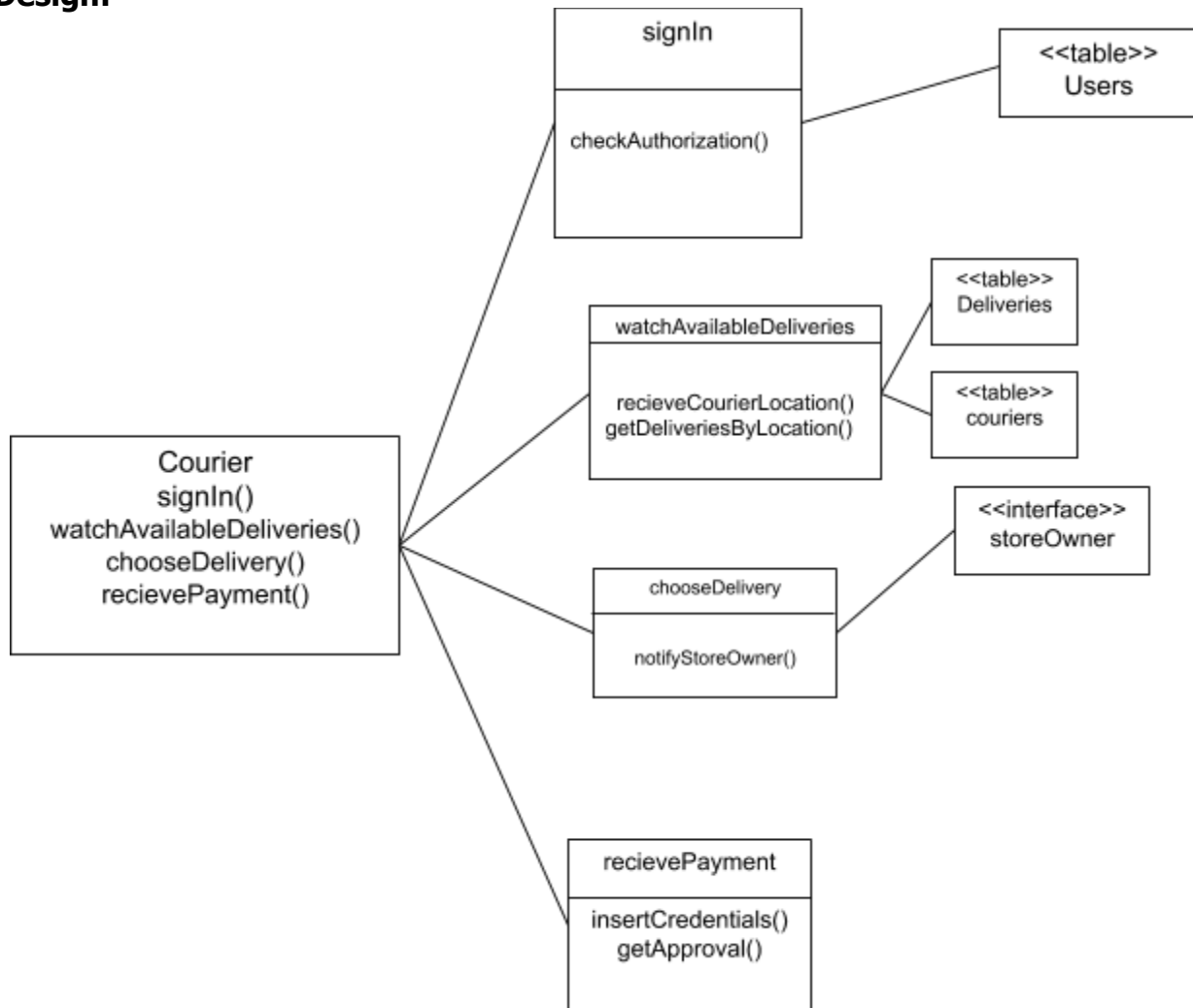
signIn()

watch Available Deliveries()

chooseDelivery()

receivePayment()

Unit Design:



user Table-

contain a list of all the business using the app

deliveries Table-

contain a list of all the deliveries ordered in the app

couriers Table-

contain a list of all the couriers using the app

business owner interface

3.3

We wanted to create a platform that reaches a maximum number of users, with a minimum expense, so we decided to create an application and not a computer program or web. The platform needs to be supported on all types of phones (iPhone, Android) so we thought of using a Flutter that in one base code we get an app that is supported on all types of phones. Due to established acquaintance with Android Studio we decided to develop the app for Android. Regarding the server, we thought of working with firebase which gives real time db in the connection, however on the recommendation of the mentor we will work with node express and mongo db.

4. Data Design

4.1. *business Table Database*

This database contains four tables. Separating the user and courier tables out allows unique storage for the basic user with multiple couriers and/or deliveries supported. The Query feature of the database management system is utilized extensively by the program.

4.1.1. User Table

Field	Type	Description
UserID	Integer	Primary Key
Name	Text	business Name
Phone	Integer	Phone number

4.1.2. Couriers Table

Field	Type	Description
CourierID	Integer	Primary Key
Courier-Last	Text	Last name of an courier
Courier-First	Text	First name of an courier
Phone	Integer	Phone number

4.1.3. deliveries Table

Field	Type	Description
DeliveriesID	Integer	Primary Key
CourierID	Integer	Who is the Courier
UserID	Integer	What business does it belong to

4.1.4. Payment Table

Field	Type	Description
PaymentID	Integer	Primary Key
Amount	Integer	Payment amount
UserID	Integer	Who pay
CourierID	Integer	Who get payment

Couriers Database

This database contains three tables.

4.1.5 User Table

Field	Type	Description
UserID	Integer	Primary Key
Name	Text	business Name
Phone	Integer	Phone number

4.1.6 Couriers Table

Field	Type	Description
CourierID	Integer	Primary Key
Courier-Last	Text	Last name of an courier
Courier-First	Text	First name of an courier
Phone	Integer	Phone number

4.1.7 deliveries Table

Field	Type	Description
DeliveriesID	Integer	Primary Key
CourierID	Integer	Who is the Courier
UserID	Integer	What business does it belong to

4.2.

business

Name: business

Description: This is the primary entrance to the system for business owner.

functions:

- signIn()
- createNewDelivery()
- retrieveDeliveryHistory()
- makePayment()

courier

Name: courier

Description: This is the primary entrance to the system for courier

functions:

- signIn()
- watch Available Deliveries()
- chooseDelivery()
- receivePayment()

5.0. Component Design

Business Component:

Description: This class collects the functionalities of the interface of business.

Operations:

Name: signIn()

Arguments: name(email), password.

Returns: boolean/void

Description: will authorize log in

Flow of Events:

1. Business owner enter FDelivery app
2. checkAuthorization() is being called, users email and password are being checked
3. user is logged in successfully

Name: createNewDelivery()

Arguments: optional only - data, time , address, city, client details(name,phone)

Returns: boolean

Description: the user fills the needed details to post a delivery

Flow of Events:

1. store owner wants to post new delivery
2. fill_details being called to retrieve delivery details from store owner
3. post delivery to repository, delivery now available to couriers to pick
4. store delivery details in DB

Name: retrieveDeliveryHistory()

Arguments: optional only - From date to date, price range, delivery area

Returns: table from DB

Description: use filters for wanted retrieve of Delivery history from DB

Flow of Events:

1. Store owner want to review his delivery history
2. Query DB to delivery history according to store owner details
3. store owner can see a table with his delivery history
4. The table can be sorted by user needs.

Name: makePayment()

Arguments: order number

Returns: interface with external system

Description: enter the order number the user want to pay for, for safety reasons it is transferred to external system

Flow of Events:

1. Courier needs to get paid for his services
2. insertCredentials() to get credit Card/Bank account details
3. getApproval() to complete the transaction

3.1.2 signIn()---->checkAuthorization()

Name: checkAuthorization()

Arguments: name(email), password.

Returns: boolean

Description: Cross-references the information and verifies the user's information

createNewDelivery()--->filledetails()

Name: filledetails()

Arguments: delivery details

Returns: boolean /void /message

Description: in order to order delivery the detailed need to be filled

createNewDelivery()--->post/create delivery()

Name: post/create delivery()

Arguments: filledetails() Returns

Returns: boolean /void /message

Description: this send/post the couriers the order

retrieveDeliveryHistory()--->delivery history query()

Name: delivery history query()

Arguments: optional only - From date to date, price range, delivery area

Returns: DB table

Description: use filters for wanted retrieve of Delivery history from DB

makePayment()--->enter credential()

Name: enter credential()

Arguments: order number (unique id)

Returns: interface with external system

Description: enter the order number the user want to pay for, for safety reasons it is transferred to external system

makePayment()--->get approval()

Name: get approval()

Arguments: enter credential() Returns

Returns: boolean

Description: returns boolean if the payment was approved or not

Courier Component:

Description: This class collects the functionalities of the interface of courier.

Operations:

Name: signIn()

Arguments: name(email), password.

Returns: boolean/void

Description: will authorize log in

Flow of Events:

1. Courier enter FDelivery app
2. checkAuthorization() is being called, users email and password are being checked
3. user is logged in successfully

Name: watch Available Deliveries()

Arguments: optional- data, area

Returns: list from DB

Description: base on the arguments return list from DB

Flow of Events:

1. Courier wants to see a list of available deliveries.

2. if the system doesn't have the courier`s location, the courier is asked to insert his current location.
3. A query to DB is performed and the results will show up as a table.
4. Now the courier can choose which delivery he would like to take care of.

Name: chooseDelivery()

Arguments: Returns of watch Available Deliveries()

Returns: boolean/message

Description: base on the arguments return list from DB courier will chose the delivery he wants to to execute

Flow of Events:

1. The Courier chose a delivery
2. Store owner need to be informed that his delivery needs to be ready to pick up
3. The delivery will be associated with this courier until the treatment will finish

Name: receivePayment()

Arguments: order number, bank account details

Returns: information from interface with external system

Description: after the business owner pass the payment of the delivery the courier receivers the money

Flow of Events:

Courier has chosen to take the delivery

Store owner need to get paid and pay the courier for his service

insertCredentials() to get credit Card details

getApproval() to complete the transaction

signIn()---->checkAuthorization()

Name: checkAuthorization()

Arguments: name(email), password.

Returns: boolean

Description:Cross-references the information and verifies the user's information

watch Available Deliveries()---->receive courier location()

Name:receive courier location()

Arguments: optional- courier id , location
Returns: message /boolean
Description: change the courier location in the DB

watch Available Deliveries()---->get delivery by location()
Name: get delivery by location()
Arguments: return of receive courier location()
Returns: list of relevant available deliveries
Description: Filter the list of deliveries to suit the location of the courier

chooseDelivery()---->notify store owner()
Name: notify store owner()
Arguments: non
Returns: boolean
Description: let the store know that a courier is executing its delivery.

receivePayment()---->insert credential()
Name: insert credential()
Arguments: bank account details, id
Returns: boolean
Description: after a courier executes a delivery he needs to get paid.

receivePayment()---->get approval()

Name: get approval()
Arguments: enter credential() Returns
Returns: boolean
Description: returns boolean if the payment was received or not

6.0. User Interface Design

6.1 Store owner user Interface

This User interface will feature the logo of the FDelivery as background.

There will be a welcoming message with the login/register option.

After successful register/login the store owner will be transferred to the main page.

Main page will include three large buttons in the middle.

The three buttons will be labeled "new Delivery", "Delivery history",

and "Delivery status" - will be disabled when there is no delivery "in the air".

- When the "new delivery" button is pressed, a new Activity will show up to retrieve delivery details such as weight, destination, origin, price ,etc.

Those details will be inserted to DB, and the new delivery will be now available for a courier to pick.

- Click on the "Delivery History" will open a new activity with current user delivery history after querying it from DB.
- "Delivery status" button will be displayed only after creating a new delivery and until the store owner makes the payment to the courier and marks the delivery as completed.

pressing this button will retrieve: delivery current status, courier details, delivery details, eta, etc.

Courier User Interface

This User interface will feature the logo of the FDelivery as background.

There will be a welcoming message with login/register option.

After successful register/login the store owner will be transferred to the main page.

Main page will include two large buttons in the middle.

The three buttons will be labeled "Available Deliveries", "Delivery history", and "Delivery status" - will be disabled when there is no delivery "in the air".

- pressing the "Available deliveries" button will open a new Activity containing a table with all available deliveries according to couriers location.
pressing on a specific item in the table will show its details and the option to take this delivery.
- Click on the "Delivery History" will open a new activity with current user delivery history after querying it from DB.
- "Delivery status" button will be displayed only after accepting a delivery and until receiving the payment.
- pressing this button will retrieve:delivery details and an option to inform the store owner that the delivery has reached its destination.

זווית הראיה של בעל העסק :

הזמן משלוח



עדכון פרופיל עסק

זווית הראיה של בעל העסק לאחר לחיצה על "הזמן משלוח" :


אנא מלא פרטים שיוצגו לשליח שיבצע את המשלוח :

שם לקוח: _____

כתובת למשלוח: _____

טלפון לקוח: _____

תאריך ביצוע המשלוח: _____



שעות לביצוע המשלוח: מ _____ עד _____ : _____

הערות מיוחדות: _____

תשלום על המשלוח: _____ ש"ח

ביצוע הזמנת משלוח

זווית הראיה של שליח פוטנציאלי :

משלוח חדש נוסף:

איסוף חבילה מפה :

שם העסק: _____


כתובת העסק: _____

טלפון העסק: _____

תאריך לביצוע: _____

שעות לביצוע: מ _____ עד _____

הערות מיוחדות: _____


קבל משלוח 

מסירת החבילה לפה:

כתובת למשלוח: _____

שעות לביצוע: מ _____ עד _____

תשלום עבור המשלוח :
000 ₪

סרב משלוח 

שבת	א'	ב'	ג'	ד'	ה'	ו'
5	4	3	2	1		
12	11	10	9	8	7	6
19	18	17	16	15	14	13
26	25	24	23	22	21	20
	31	30	29	28	27	

7. REQUIREMENTS MATRIX

system components	Description	page
INTRODUCTION	overview of this document, description, purpose and scope of the software.	1
SYSTEM OVERVIEW	general description of the functionality, context and design project.	4
SYSTEM ARCHITECTURE	diagram showing the major subsystems and data repositories and their interconnections.	5
DATA DESIGN	Data Description into data structures.	9
COMPONENT DESIGN	closer look at what each component does in a more systematic way.	12
HUMAN INTERFACE DESIGN	Overview of User Interface, Screen Images	17