THE MINIPROJECT

ENTERTAINMENT RESORT

MANAGEMENT SYSTEM

AMIT "DIKKAR"

AMRUTA "AMU" GANDHI

OMKAR "OMI" EKBOTE

LIFECYCLE

"Those Software Engineering concepts aren't to be applied bottom-up!"

2 weeks of sleepless nights, incessantly occupied days and more research than we've ever done in the entire semester: it was all wrapped up in this fostered time period, jumping from varied versions of code, interface, design and schema. Change is always drastic and it triggers a nasty chain reaction: you make one tiny change somewhere and everything has to change, to adopt and assimilate. For one fine moment its all working and at the next keystroke, everything seems broken. Well, at the end of those seamless image searches and perpetual drop-create routines, we figured out *one* thing: those Software Engineering concepts aren't to be applied bottom-up.

Define scope » Analyze/Design » Code » Test/Validate

What everyone did was, ironically,

Pick a topic » Muster up an ER Diagram » Design GUI » Design Tables » Re-design GUI » Change scope » Re-design tables » Code » More code » Change GUI » Finalize tables » Test » Debug » Redo the EER

Now, if you're reading this already having done all that, have a laugh! If you're yet to jump through the loops, do it right and have the last laugh!

The overview of the flow of our (mini)-project (we keep saying that) is briefly illustrated by a flow diagram. This was the one thing we did at the right time: *before* starting with the interface design. It defines a scope, gives a clear understanding of user sessions for all 3 types of users (and looks good)! An upheld realization would be, the more Software Engineering diagrams you draw at the design stage, the better is your understanding about what you want to achieve. At the end of this document, we'll disclose the list of software we leveraged to pull it off. It's no secret: the quality of a product lies in the workman's skills and not the tools used: although, some great intuitive tools do boost some productivity! Another word of experience: explore your possibilities before you define your limits.

"An upheld realization would be, the more Software Engineering diagrams you draw *at the design stage*, the better is your understanding about what you want to achieve."

DEPLOYMENT

"You can't just skip testing; it's inevitable, unless you want to be embarrassed live."

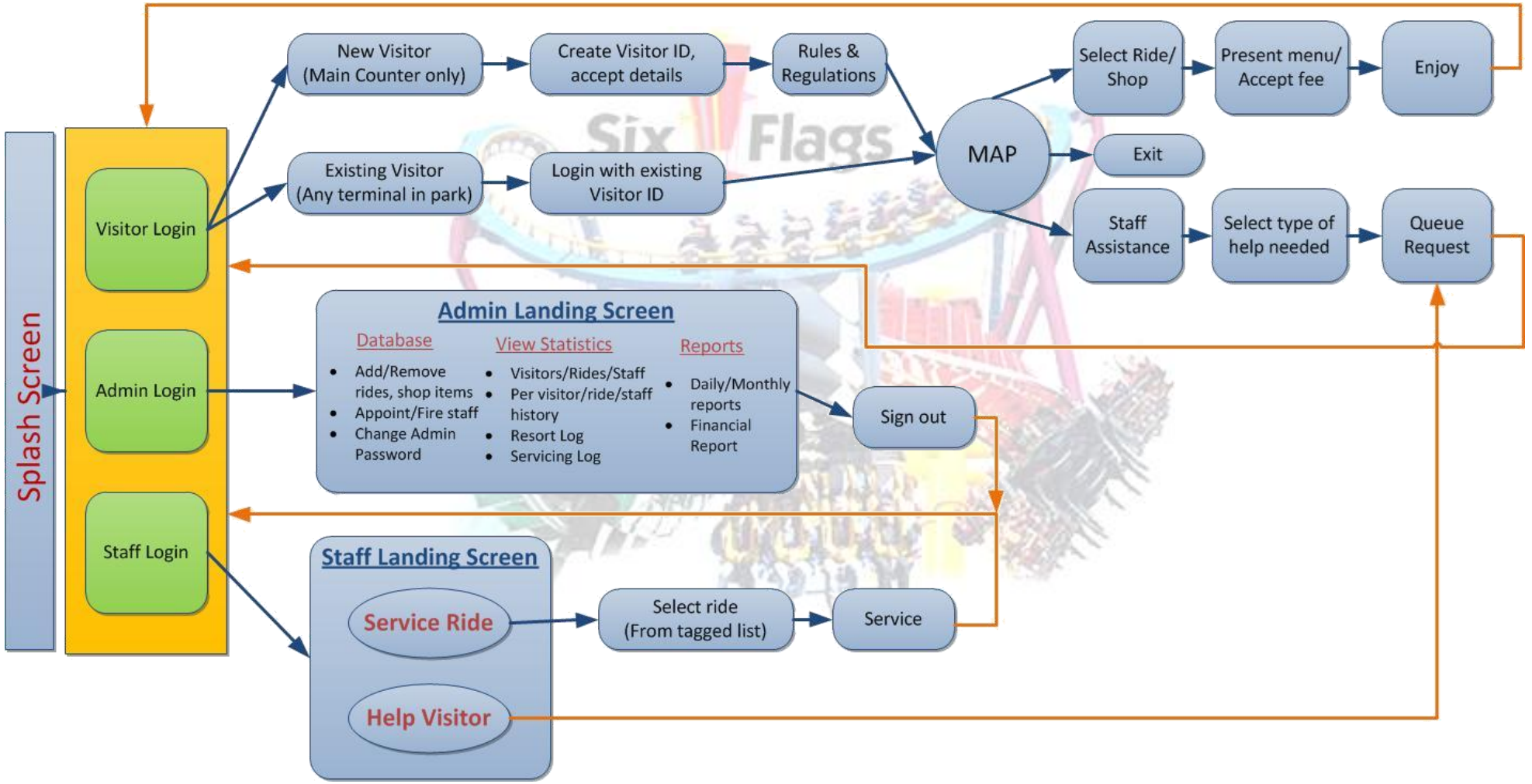
When you make something in a group, everyone wants to do everything. Given the constraints of 20th century software ('nough sworn), collaboration doesn't *just happen*. Whilst the making, you'll end up with multiple versions on multiple systems with no idea about which one contains *all* features in the end! You'll have to pick one. Versioning is like the weather report: you can write a whole book about what, when & why *after* its all over, but you don't have a damn clue when to end a version and where to start a new one *whilst* you're at it! Two cruel facts about these development software: The VB project only works in the system you made it on; and Oracle® wasn't made to be portable. At some point, when you'll grill through hell to find and use those IMP and EXP binaries, you'll be sufficed by their gruesomeness. In this agony is where the deployment begins.

Software isn't made for oneself: it's for everybody! But it's a lot of pain to make it work for everybody. What's the big deal about abstraction, you ask? This is it. When they make clothes, they've no idea on whose physique they'd be flaunted. This is something like that, except software doesn't come in different sizes. There has to be *one* package that 'runs' everywhere. Those endless pages of testing strategies are worth a read. You can't just skip testing; it's inevitable, unless you want to be embarrassed. The good thing is, your friends are the testers: ask those scoundrels to tether your software and you'll have a temperament nastier than an official SCM report! You know, they looped those SDLC models for a reason: it never ends.

[TECHNICAL BLABBER]This deployment is a standalone application that works on any machine: you don't need Oracle® nor do you need to configure anything manually. Its been ported to be truly autonomous, with sample data pre-loaded into the database for an easy demonstration. If you're reading this then you've probably voyaged through the Setup and we'd like you to know that you can always remove all traces of this deployment from your computer by using the Add/Remove Programs in Windows® where it's listed as "Entertainment Resort". All changes you make to the database via the application will persist on your computer but can be reset by reinstalling the deployment. You can view the actual contents of the database from the Start Menu entry.

"Versioning is like the weather report: you can write a whole book about what, when & why *after* its all over, but you don't have a damn clue when to end a version and where to start a new one *whilst* you're at it!"

ROLE-PLAYING



"The overview of the flow of our (mini)-project (we keep saying that) is briefly illustrated by a flow diagram. This was the one thing we did at the right time: *before* starting with the interface design!"

IN A NUTSHELL

"So here we are, sitting in a WiFi-equipped, 300 sq.ft. lab, dreaming about a 500 acre resort in which our software will be deployed!"

The *project title* instils a sense of real-world scenarios (albeit vague) in which it is likely to be used. This ranges from what kind of screen it will be showed on (we demo on a PC, but imagine an actual deployment environment – industry standard and fancy at your will!) to the type of users who would be using it. (Un)fortunately or otherwise, our 'textbook' project topics are rugged over millions of times and you can always peek over someone else's shoulder to get an idea about how it *has been* for all these years. So here we are, sitting in a WiFi-equipped, 300 sq.ft. lab, dreaming about a 500 acre resort in which our software will be deployed. (It wasn't really a dream, as you'll see when you get to the references & software listings later)

Now here's how we visualized the Entertainment Resort Scenario: There are terminals placed at various locations in the resort. Any user (visitor/staff/administrator) can login to the database from any of these terminals.

VISITORS

- There is a Main Counter at the entrance of the resort. A visitor checks-in to the resort by filling up a form with his personal details. He is provided a (fancy-looking) receipt with his booking details on it.
- At every other terminal, he logs in by using his unique VisitorID to get to a map.

STAFF

- There are two types of staff members:
  - Assistants, who help the visitors upon their request.
  - Mechanics, who service and repair the rides tagged by the administrator.
- They can login using their username & password at any terminal and proceed to their jobs.

ADMINISTRATORS

- The administrators can open up the Landing Screen and perform tasks, such as appoint staff, add/remove rides, change prices and taxes, generate financial and other reports etc. from any terminal.

You don't need to read this, actually – it's all evident from the nifty little diagram up there!

"(Un)fortunately or otherwise, our 'textbook' project topics are rugged over millions of times and you can always peek over someone else's shoulder to get an idea about how it *has been* for all these years!"

FACTORY SETTINGS

"This being a *Demonstration Edition*, we've pre-loaded just enough stuff in the database to get the reports working properly!"

"When can I start?!", you may ask. Security is overrated, as always, since it only makes sense when you know someone's going to break in. Can you see anyone trying to break in to this 20th century software? Not till the end of *this* lab! Anyway, this being a *Demonstration Edition*, we've pre-loaded just enough stuff in the database to get the reports working properly. The whole point of this project was *database management*, so everything and every-user can be changed, removed and configured by the Administrators. To start off, we have some 30-odd visitors in the park for November'10, an assistant & a mechanic as staff members and the default admin. There are 4-5 rides of each type, the shops & restaurants are filled with toys and food, taxes and entry fees are preset. This thing is ready for a test ride! Now here's where the key goes in:

VISITORS	
Date of Visit	**VisitorIDs**
1/Nov/2010	1,2,…,5
2/Nov/2010	6,7,…,10
3/Nov/2010	11,12,…15
--	--
Kids (Age<15)	3,7,10,12,…
Adults (Age>=15)	1,2,4,5,…
--	--

STAFF		
Username	**Password**	**Role**
amit	pass	Mechanic
omkar	pass	Assistant

ADMINISTRATOR	
Username	**Password**
admin	oracle

When the project starts off, it initializes to today's date. So there's a special feature we had to add to the Admin Landing, to change the current date and time. Even though it violates the laws of physics, it's the only way you can test validations!

Admin Login » Realtime Settings » Change Current Date (Ctrl+D) » Set date as per above table » Logout

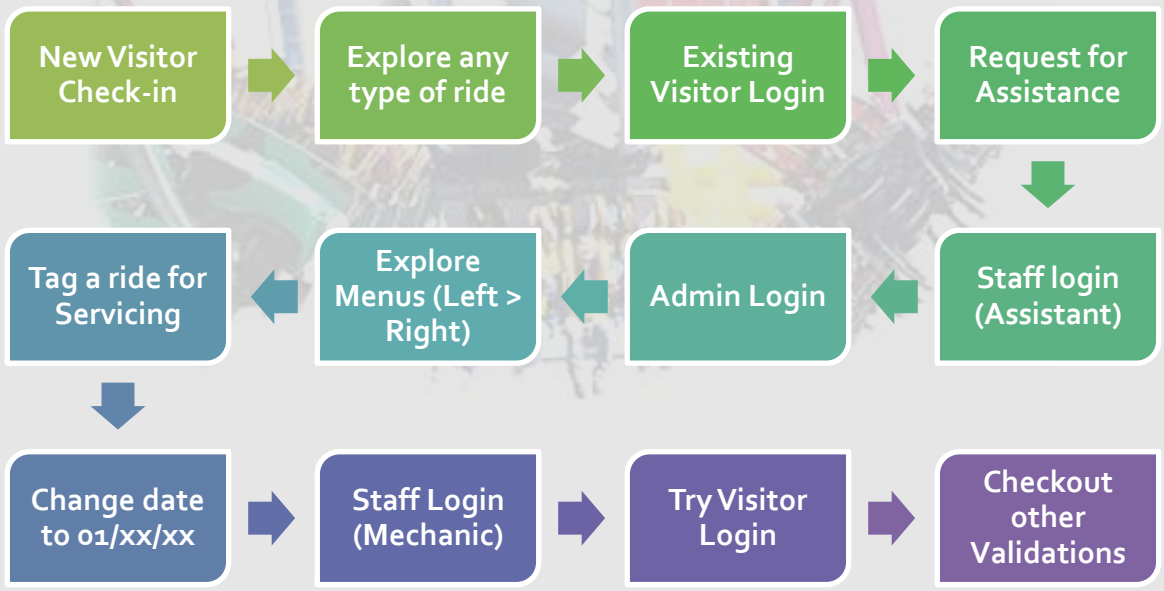
"The whole point of this project was *database management*, so everything and every-user can be changed, removed and configured by the Administrators."

GUIDED TOUR

"Ah, screw it – 'we did what we can and the deadline struck.'"

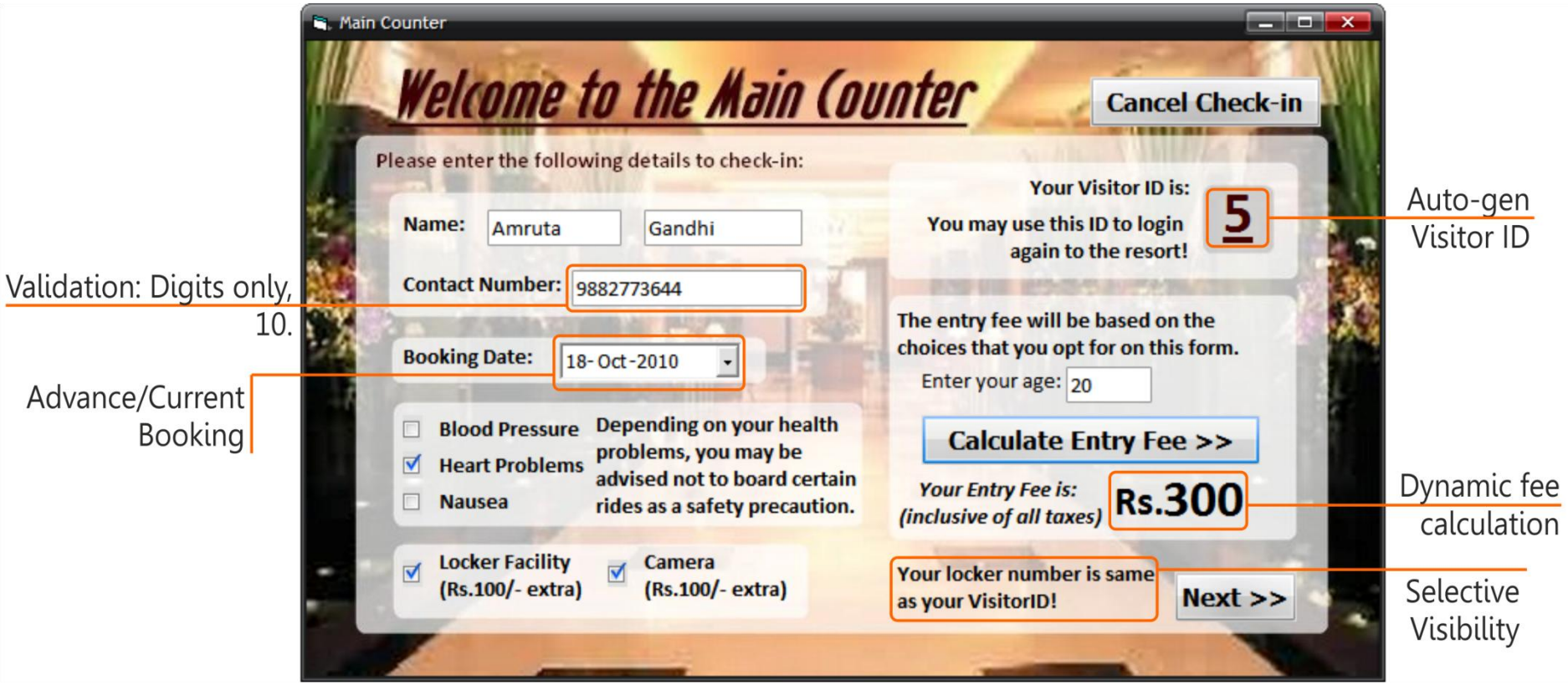
When you present something new to a meek personality, it's a pleasure to watch them toy with it, explore around and be amazed! Feel free to do that with this application: it's rugged enough in the right places, but you may have to be careful with the Admin Console. You see, validation & robustness is nothing better than dotting the i's and dashing the t's: last-minute and condescending. Everything works by now, all features implemented and it'd 1am on the clock: who'd be keen enough to make sure all goes well in the hands of a 5-yr-old user and doesn't succumb to his deranged inputs? Ah, screw it – 'we did what we can and the deadline struck.' – a textbook excuse to the rescue!

For those who want the rekindling pleasure of holding someone's finger and walk around in contention, here goes:



"Validation & robustness is nothing better than dotting the i's and dashing the t's: last-minute and condescending."

MAJOR ATTRACTIONS



These windows are the crux of the 'notable annotations' of our project. They aren't the *best* screens, they are the *essentials*. It's not like, if you've seen this, you've seen it all. Of course, there's more to it than what meets the eye: the code speaks out features and not all of 'em make it to the frontend. There's a lot going on behind the scenes and that's where all the action is!

"It's not like, if you've seen this, you've seen it all! There's more to it than what meets the eye."

MAJOR ATTRACTIONS

Advertisers like crowds a lot, but PR guys hate disasters! If the former have a reason to make guests enjoy more rides, the latter have a reason for guests to stay off some rides! The constraint is pretty straightforward here – there's a one-to-one correspondence between ride specifications & visitor profiles. The complexity, if any, is all in the code! There are several such checkpoints for various anomalies in the project: there's a daily visitor limit, a fix for the book-my-ticket-in-the-past syndrome and all these can be demonstrated by being a bad user and trying it out in the field!

"If advertisers have a reason to make guests enjoy more rides, then PR personnel have a reason for guests to stay off some rides!"

MAJOR ATTRACTIONS

Looking around in the lab, at other projects and the people working on them gives you ideas. Help them, and it gives you experience. Combine that, and you'll end up incorporating bits and pieces from tons of projects in your own application!

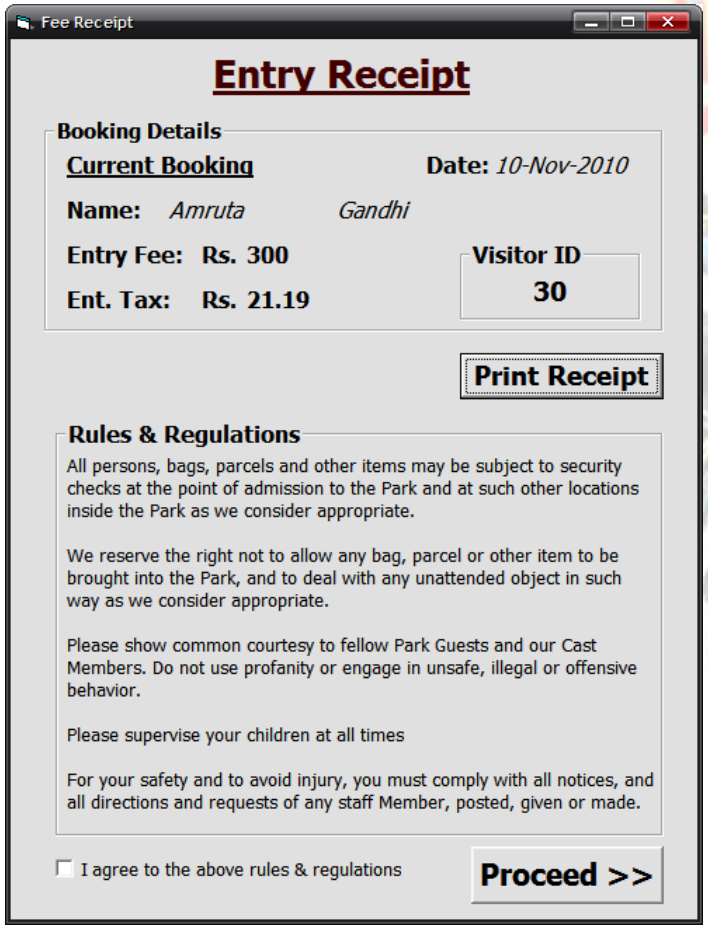
This – the apparent catering window – was one of the full-fledged inventory management projects. It works just like that, but with reduced logging activity. There's a cart, to which you can add as well as toss out items, an automatic total, grouped menu – all the fuzz at front. Behind all that, only the cost & selling prices are logged. If you haven't noticed yet, this is an obsessively 'feel-good' project!

"Looking around in the lab, at other projects and the people working on them gives you ideas. Help them, and it gives you experience."

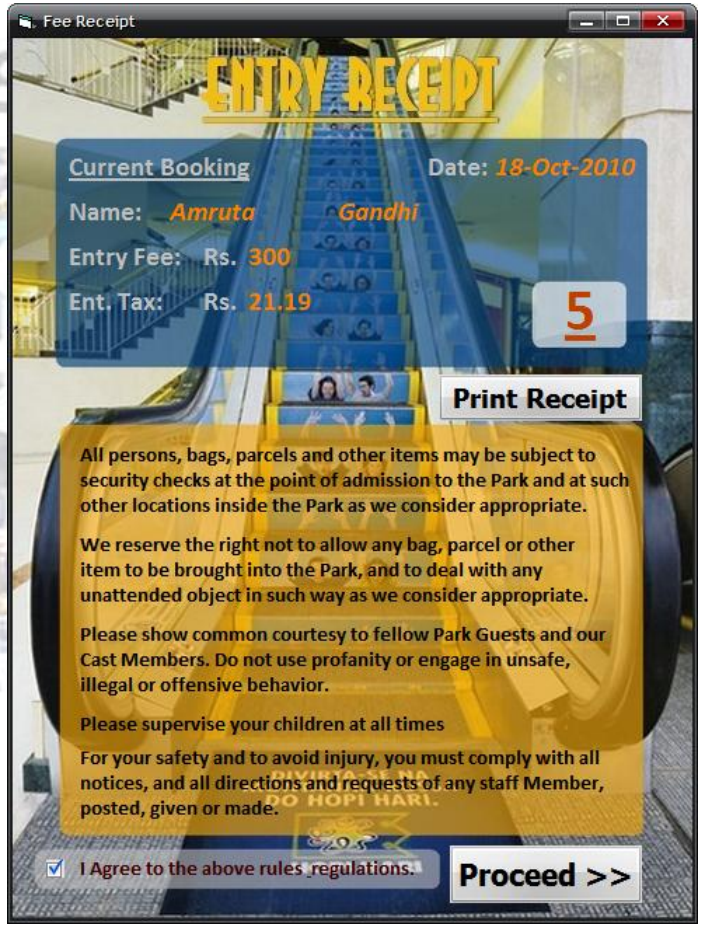
MAKEOVERS

Like the story of every damsel in distress that hails to the city in search of fame; the pepped up face of this project wasn't always that glamorous! Here's a few of the intense makeovers:

BEFORE



AFTER



"If it looks pretty, you have 10 points over everyone else."

MAKEOVERS

BEFORE

AFTER

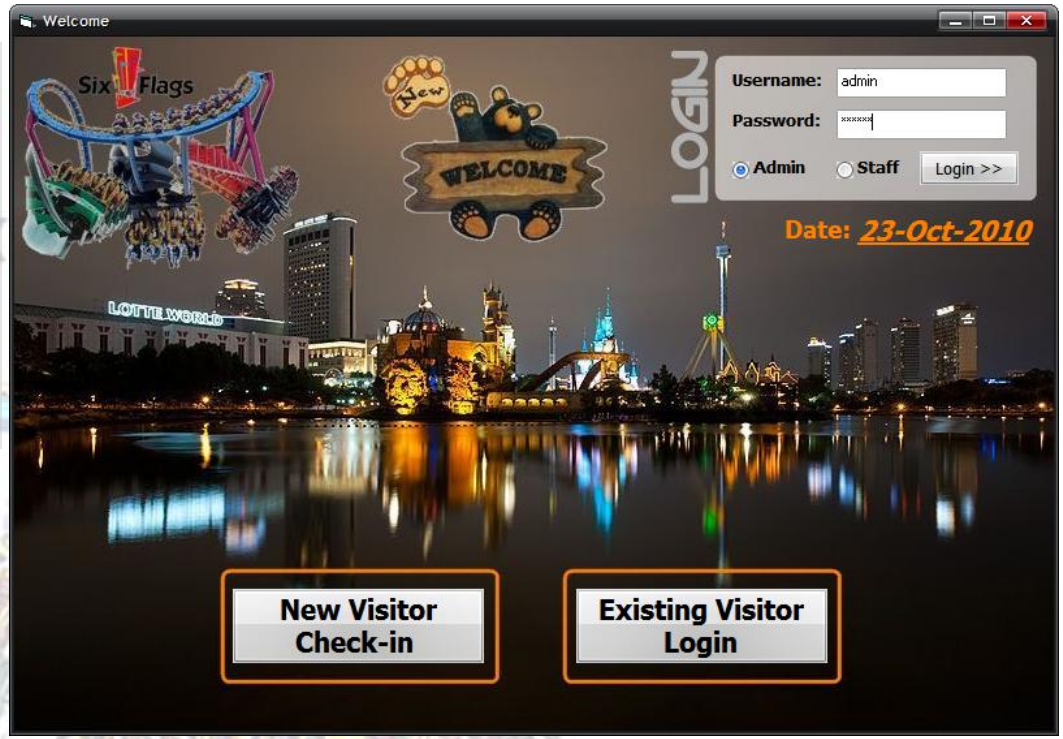
"First the code, then the colour. Because everything changes."

MAKEOVERS

The crazy thing about makeovers is that you have to know, firsthand, *what* there is, to make over! It's a last-minute job (well, what isn't?) which gives you the triumph of *completion*.

To be honest, every single window was changed. In fact, the more you realize you can achieve, the deeper you're pushed into *improving* what you already have. Don't do that.

BEFORE



AFTER

You have to stop at some point, and the earlier you see the stop sign, less bumpier would be the ride! We worked till the midnight of the day before our final demonstration. Some worked till minutes before. And it really doesn't get better after a certain threshold. It crashes.

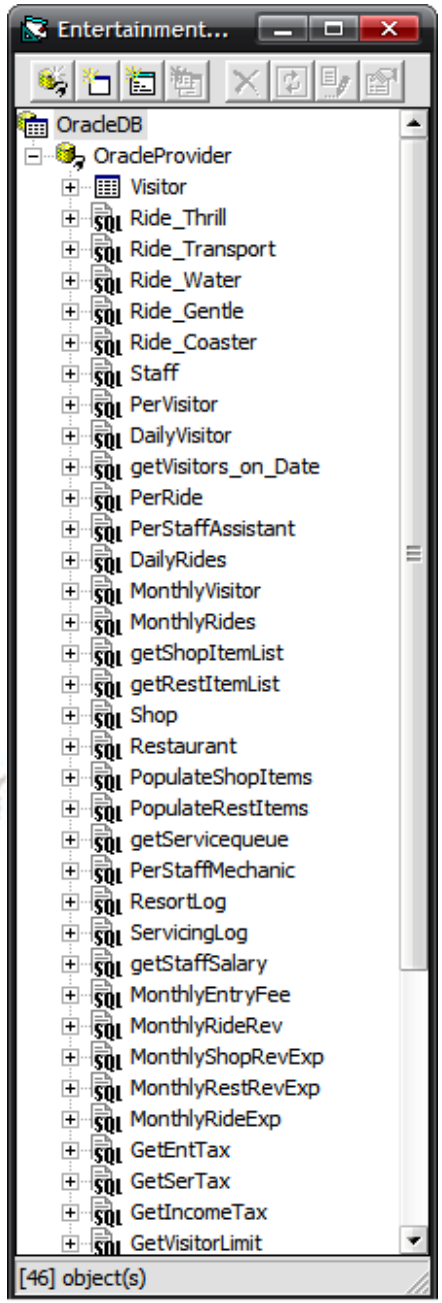
"You have to stop at some point, and the earlier you see the stop sign, less bumpier would be the ride!"

SOURCE CODE

There's a lot more to it than dragging controls onto forms. It's all about what's under the hood! And we rarely write code, we always just reuse (to put it mildly). Here, let us classify it for you:

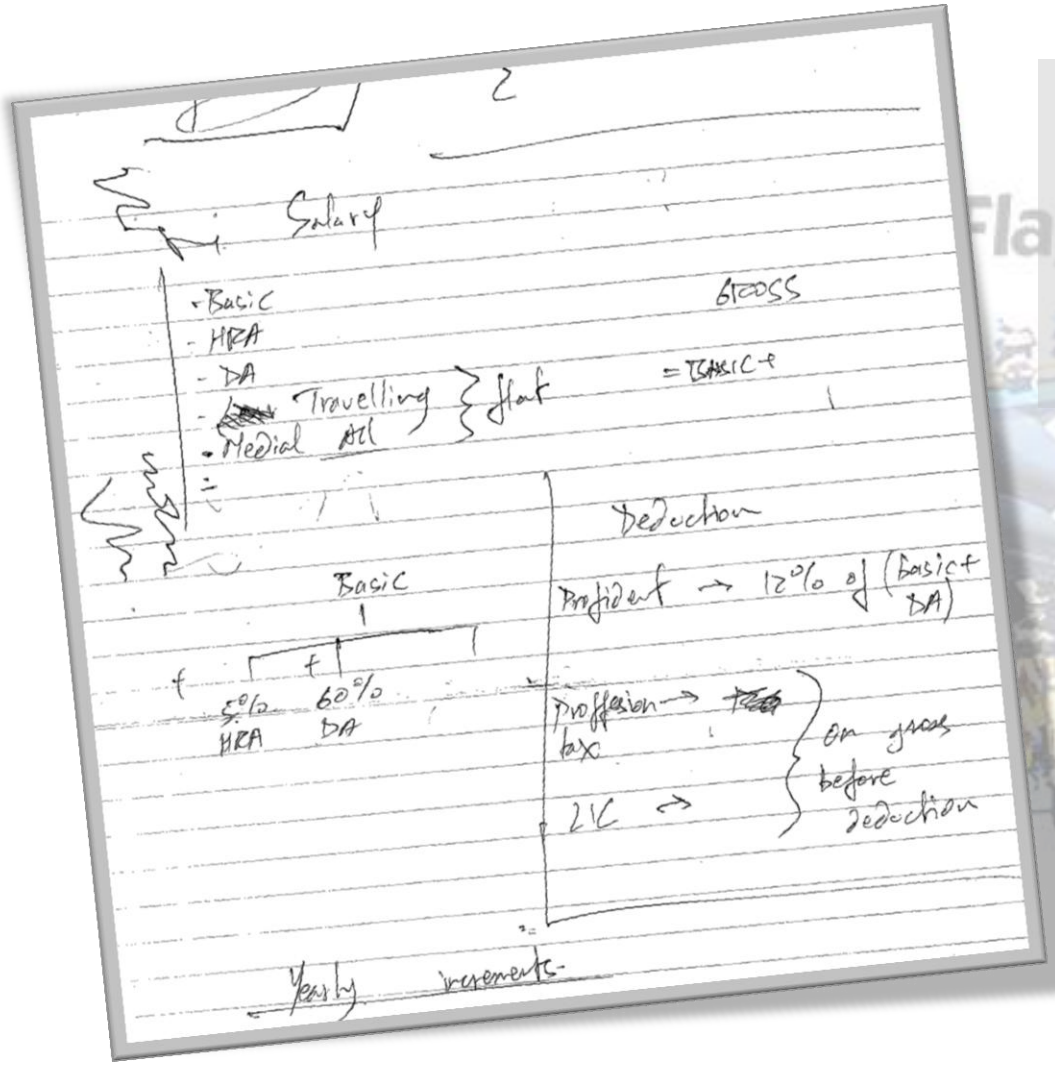
entertainmentResort.vbp	
Code	**Form**
Simple login without password checking	visitor_existing.frm
Login with password	welcome.frm
Date Checking (particular date, before/after date)	welcome.frm maincounter.frm
Auto-generate VisitorID	maincounter.frm
Populate ListBox with items from database	restaurant.frm Staff_Mechanic_Landing.frm
Initialize certain (global) variables with values from database (e.g. settings)	frmSplash.frm
Mathematical Calculations	Admin_MonthlyFinance.frm
Tax Calculations	Rules_Reg.frm
VB ←→ Oracle date format resolution	Admin_ChangeDate.frm
Parameter passing to DataEnvironment Commands	Admin_DailyRides.frm Admin_MonthlyVisitor.frm
Dynamic text labels in data reports	Report_PerRide.dsr
Change user authentication in database	Admin_ChangeAdminPass.frm

(re)Use it wisely! And don't forget to refer the indigenous SQL queries in the Data Environment Commands!

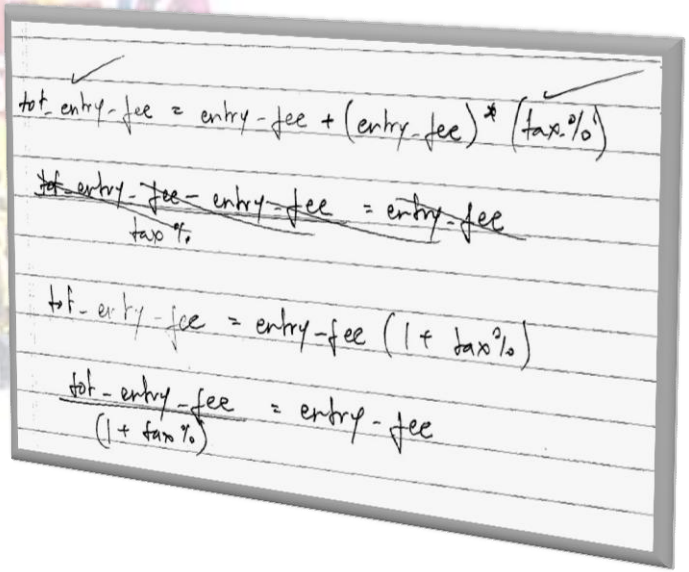


"There's a lot more to it than dragging controls onto forms. It's all about what's under the hood!"

THE MAKING

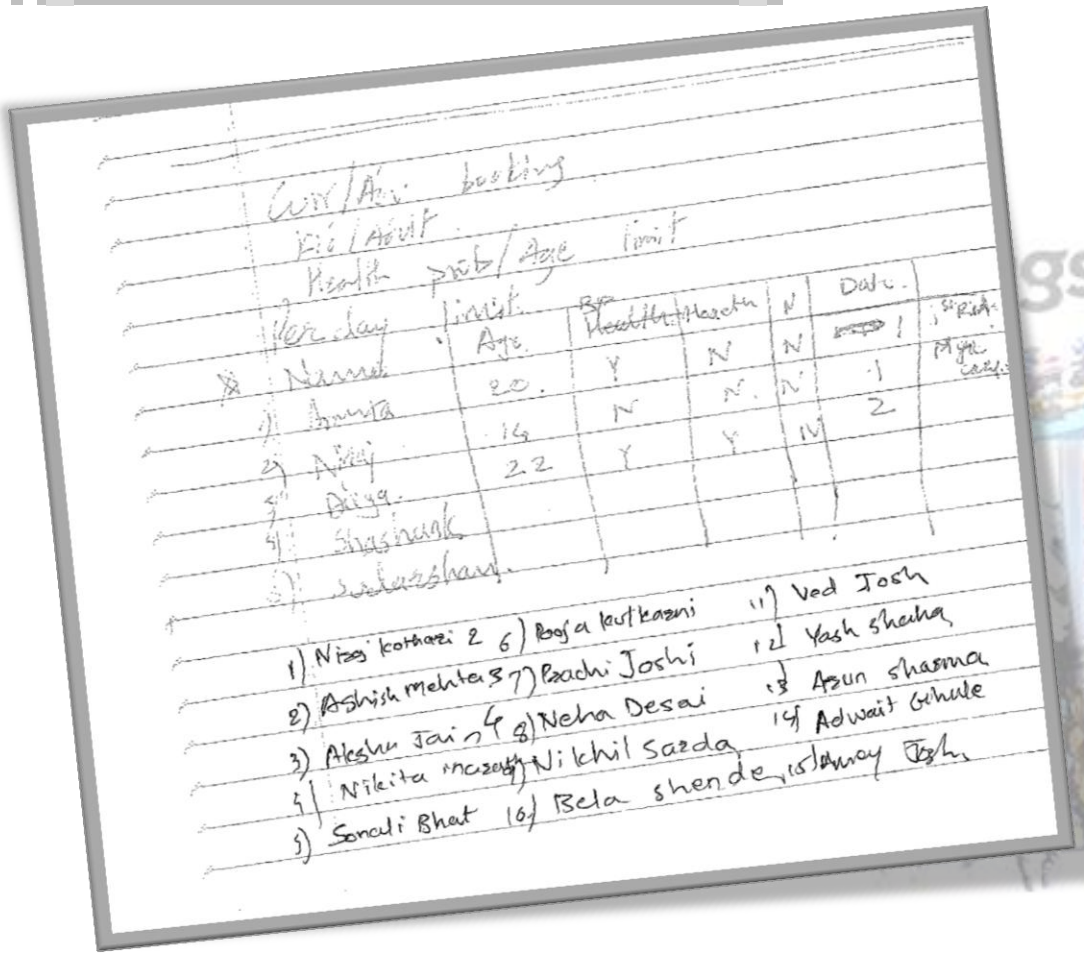


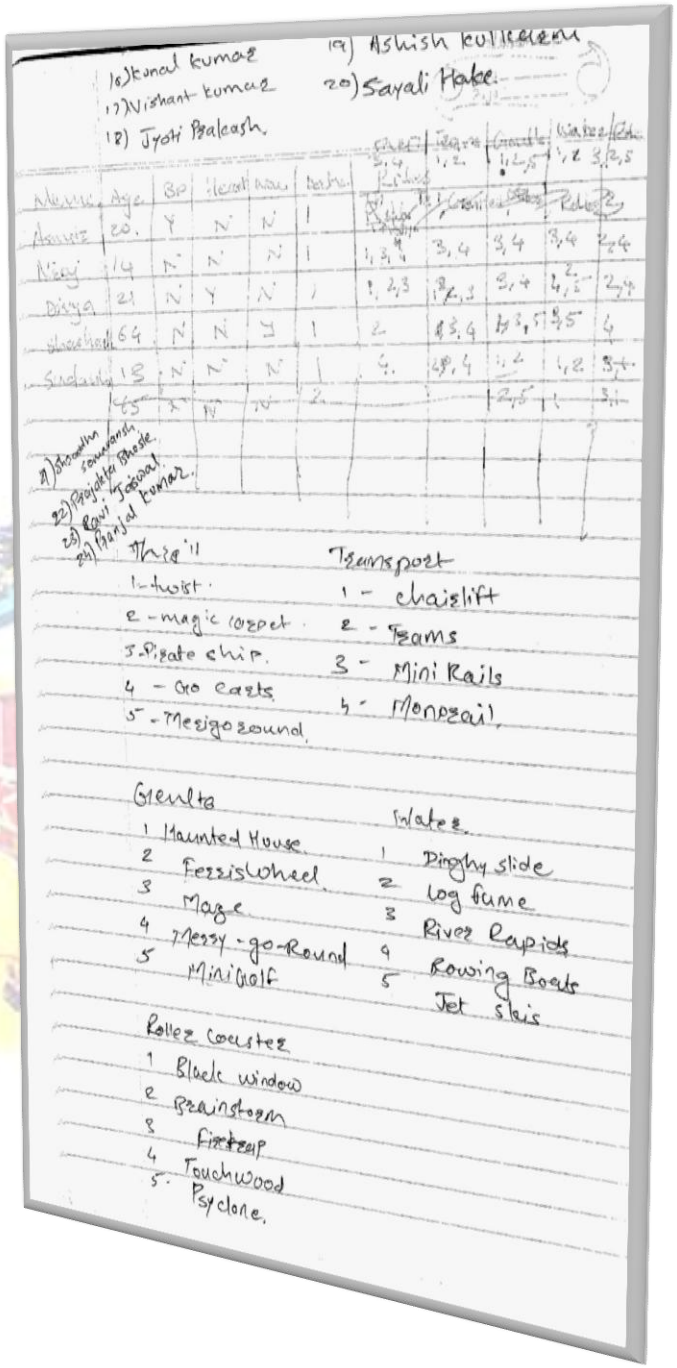
Authenticity is a virtue: if you're developing a project pertaining to a real-world scenario and you want to stick to [the reality], research is inevitable. We consulted a CA intern for the financial reports: the components and format. But there was some homebrew research too: that entertainment tax derivation was the paramount of our mathematical abilities: and you don't need to clear M3 to pull it off!



"Authenticity is a virtue: if you're developing a project pertaining to a real-world scenario and you want to stick to [the reality], research is inevitable!"

THE MAKING

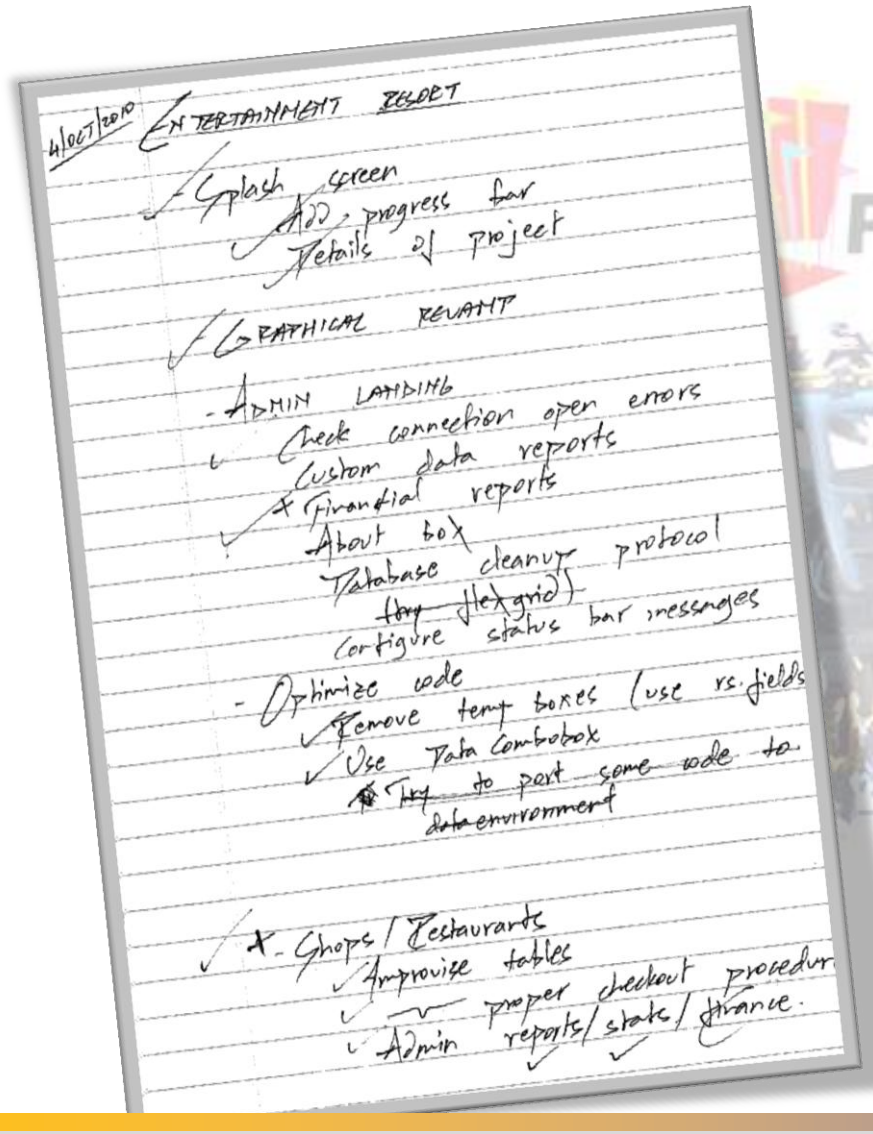


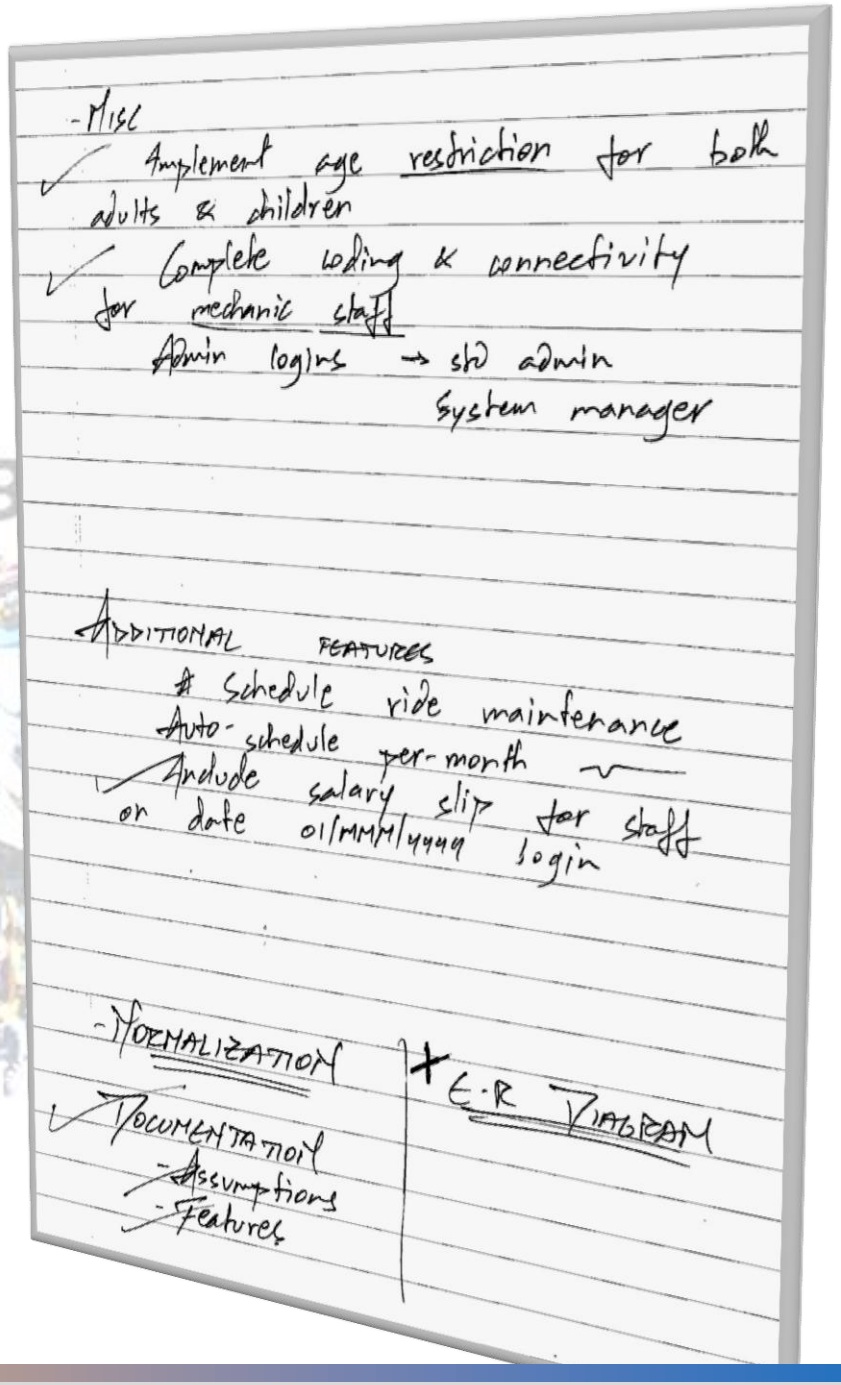


Sample data is the agonising truth of nurturing your baby. True, having it in the first place was a lot of fun, but then you have to dress it up, give it love, care and make up stories about how it came here. Interesting stories.

"We used the attendance roll that was tying in the lab for visitors' names; so if you find any familiar ones, it's not just coincidence!"

THE MAKING





"To-Do lists: oh, so many to-do lists! The perpetual indecencies of sleepless nights... Lullabies for some, nightmares for others!"

REFERENCES

"The quality of a product lies in the workman's skills and not the tools used!"

As promised, we'll reveal the software and other resources we leveraged to ensue this project – it's no secret, actually. Despite the extinct development tools at the backend, the cosmetics have been latest and chic. Here's what made the difference:

SOFTWARE		
Product	**Usage**	**Description**
Microsoft® Visual Studio® 6.0	IDE and deployment	Abide by the syllabus, just like we abide by the driving rules.
Oracle® 9i	Database Management	This version onwards, the Oracle Provider for ODBC is available which resolves some access permissions and locking issues.
Microsoft® Visio® 2010	EER Diagram Flow Diagram	Amazingly intuitive modelling software. Both diagrams were finished in 10 minutes each (not to mention the numerous versions).
Faststone® Image Viewer v4.2	Form Backgrounds & Overall GUI	Refrain from using heavy-duty productivity software. Keep it simple. It does the job.
Faststone® Capture v6.7	Screenshots	We managed to take 80+ screenshots of this thing and it wouldn't have been possible without this neat little software.
VMware® Workstation v7.0	Virtualization	Legacy software runs in legacy OS. Apart from not having to dual-boot, it helps immensely in portability.
Google® Chrome®	Web Browsing	Virtualization is resource-demanding. You need a fast, lightweight browser to work with.
Google® Image Search	Stock Images	71 stock images were downloaded. 1/3rd of them used. Leave it to the artist, even if he's choosy.

"Despite the sleepless nights, the crushing deadlines and frustrations of software and their counterparts, this has been a learning experience at best: teamwork, collaboration, SRS documentation: all of it isn't overrated, it just makes sense now!"

THANKYOU

VISIT AGAIN.






The purpose of this document is to demonstrate a mini-project titled 'Entertainment Resort Management System' and is intended to be used in conjunction with its deployment.