# Which Plane is Boarding Now? Where To?

Dillan Ganesh, Xia Yu, Nolita Ebama-Mpeta
Ali Ghoroghi, Rowan Gilmore

Supervisor:
Dr N. Dingle

# Presentation Outline

- Introduction
- Research
- Development
- Deployment
- Conclusion

# Overview

- Collaboration with Amadeus
- Develop a representation of logged airline traffic information
- Open to any implementation methods
- Parsing of log files (Rain, Ali)
- Interface / data integration (Dillan, Nolita, Rowan)

# Specification

- Extraction of check-in/boarding messages  from log files in EDIFACT format
- Provide airport identifiers, date/time and message type to front end
- Data management system to handle storage and communicate with interface
- Illustration (graphical and textual) of the information in a user interface
- Definition of a colour and thickness scale to highlight the level of traffic
- User interaction: map manipulation, date/time handling etc.

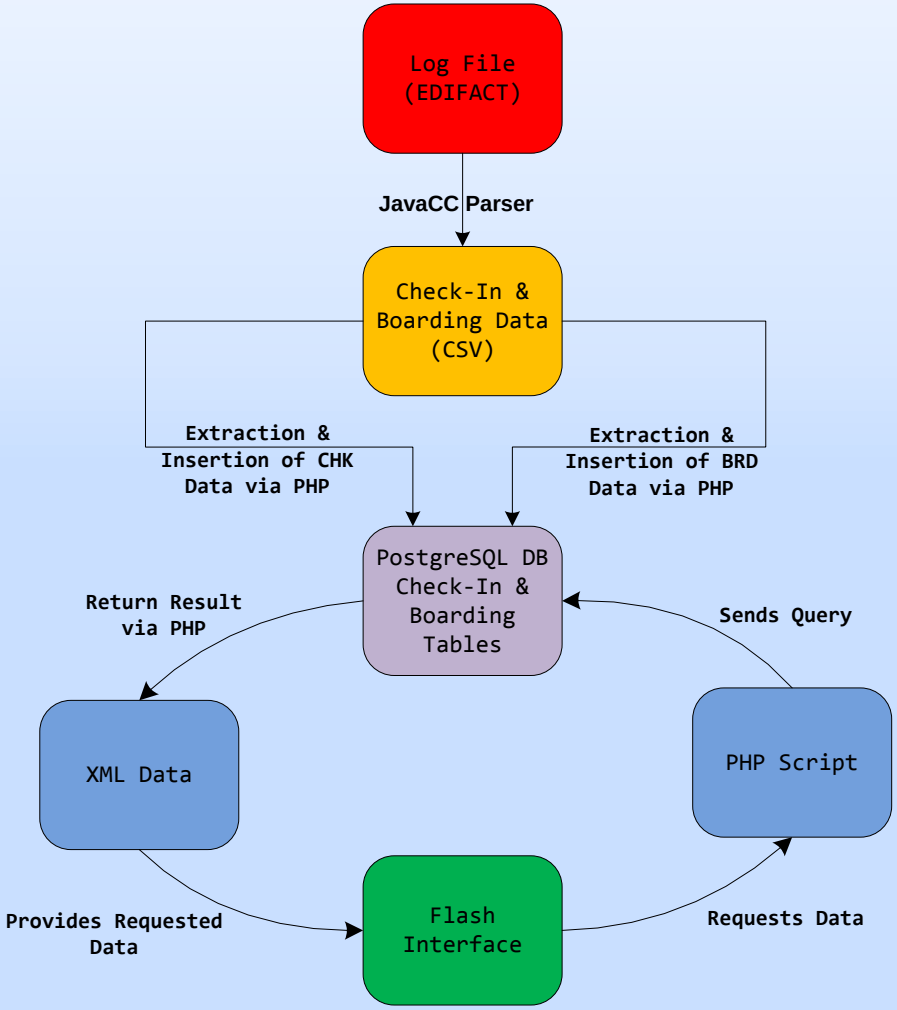
# Exploring Design Approach

- Studying the structure and elements of EDIFACT
- How can the required information be extracted?
- Parser – can analyse the sequence of tokens efficiently
- How to best illustrate global airline traffic?
- GUI – use of a well-developed map API
- Use of DBMS – reduction in back-end GUI programming effort
- Eventual deployment could be web-based or as a desktop application

# Choice of Platform

- JavaCC – frequently used parser writer with required capabilities
- Google Maps API…
  - ✓ Standard online map features – zoom, pan etc.
  - ✓ Added sophisticated functionality e.g. geocoding, 3D perspective
- Flash Maps version…
  - ✓ Object-oriented, cross-platform, easily integratable
  - ✓ Powerful IDE – efficient high-quality front-end design, good looking!

# System Integration



# Why JavaCC?

- Has a top-down methodology (recursive descent) – easier to debug
- Includes a built-in state machine
  - ✓ Flexible token matching
- Output is pure Java code…
  - ✓ Platform independence
  - ✓ Can provide better compatibility for GUI sub-group

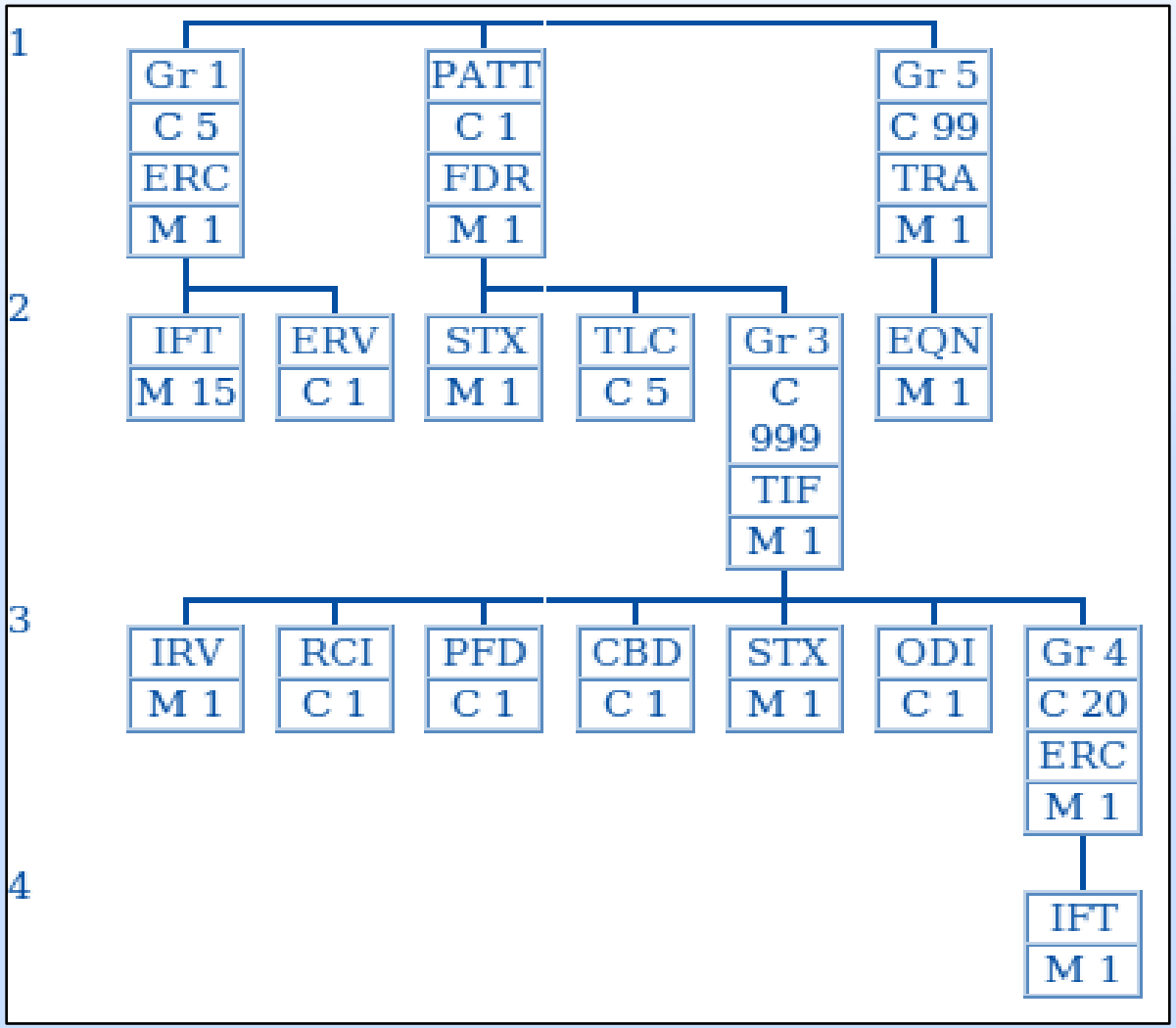
# Messages to be Parsed

- CCKIUR – check-in message
- CBDIUR / CBDAUR– boarding message
- CDBIUR / CDBAUR – de-board message
- Two linked lists for recording the check-in and boarding messages:
  - LinkedList <checkinInfo> checkinList
  - LinkedList <boardingInfo> boardingList

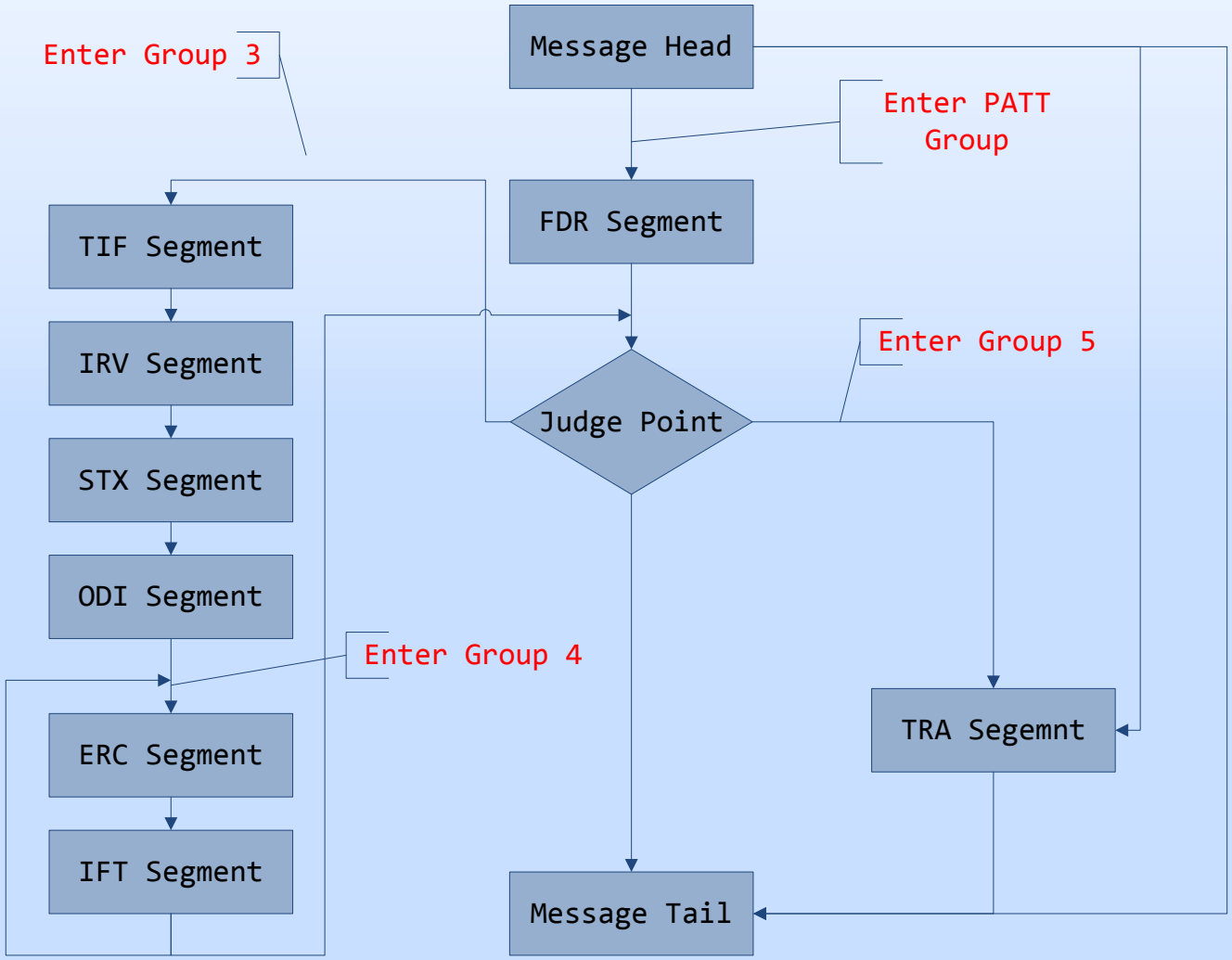
# An Example of CBDIUR

2010/01/22 08:46:47.502815 obeat015 FE26291UA0OGAVG-22749 RECEIVER: Nb=4721 Len=482
UNB+IATB:1+1ASRDC+ATCPING+100122:0846+00BYL7DK1J0001+00CHX338OQ0001++T&
UNH+1+CBDIUR:07:2:1A+000100CHX338OQ&
FDR+LJ+1+20100201+ICN+BKK&
STX+AC:OP&
TLC++GTE:405&
TIF+ZZZYXTEST:A+ATCSYD::000414B57C83CD49&
IRV++++::DID:000414B57C83AA9C&
RCI+:2CH6S5&
PFD+028F+N+2&
CBD+Y&
STX+:B&
ODI++BKK&
ERC+17351:WEC&
IFT+1::::EN+Boarding eligibility checks bypassed&
TRA+LJ+1&
EQN+2:BOO+0:BOI+0:ABT+0:IBT+2:ABJ+0:IBJ+2:JAA+0:JAI+0:AAT+0:AIT+2:JBA+0:JBI+0:BAT
+0:BIT+0:CSA+0:JMS+0:IBC&
UNT+16+1&
UNZ+1+00BYL7DK1J0001&

# Structure of CBDIUR Message



# Message Match Algorithm



# Segment Matching Example

```
/* Tokens for all the states */
<*> TOKEN : { < OTHER : ~[] > }

/* Define the head of the message */
TOKEN : { < CBDMSG : <UNH> (<OTHER>){3} <CBDHEAD> > : CBD_MSG }
TOKEN : { < #CBDHEAD : "CBDIUR"|"CBDAUR" > }

/* Define the tokens for PATT Group */
<CBD_MSG> TOKEN : { < CBDFDR : "FDR" > : FDR_SEG }
<FDR_SEG> TOKEN : { < ONPOINT : <AIRPORT_CODE> > : CBD_GROUP3 }

/* Define the tokens to match ID information for CBDIUR */
<CBD_GROUP3,CBD_GROUP3_OR_ERC> TOKEN : { < CBDTIF : "TIF" <DELIMITOR
<CBD_IRV_SEG> TOKEN : { < CBDIRV : "IRV" <DELIMITOR> > : CBD_PASSENG
<CBD_PASSENGER_ID_TYPE> TOKEN : { < CBDIDTYPE : ("UCI" | "DID") <DEL
<CBD_PASSENGER_ID> TOKEN : { < CBDID : (<ALPHABET> | <NUMBER>)+ > :

/* Define the tokens for STX Segment */
<CBD_STX> TOKEN : { < CBDSTX : "STX" (<DELIMITOR>){2} <ALPHABET> > :
```

# Performance Improvement

- Original way to run parser: java EDIParser < $logfile
- Using UNIX command grep as a filter to the raw data:
  - MSGs="CCKIUR|CBDIUR|CBDAUR|CDBIUR|CDBAUR”
  - grep -E "$MSGs" "$logfile" | java EDIParser
- The results:
  - with filter real 0m0.678s
  - without filter real 0m39.898s

# Output Format

- CSV file (check-in/boarding, date/time, airport ID(s), passenger ID):

  CHK, 2010/01/22 23:42:18, SYD, 001444A0AD17D2AF

  CHK, 2010/01/22 23:43:05, GLA, 10000000000CD47E

  CHK, 2010/01/22 23:46:57, ICN, 000424B59B40BBA6

  BRD, 2010/01/22 23:26:57, LHR, SYD, 000414B5A33F6155

  BRD, 2010/01/22 23:29:39, HKG, LHR, 000414B5A342315A

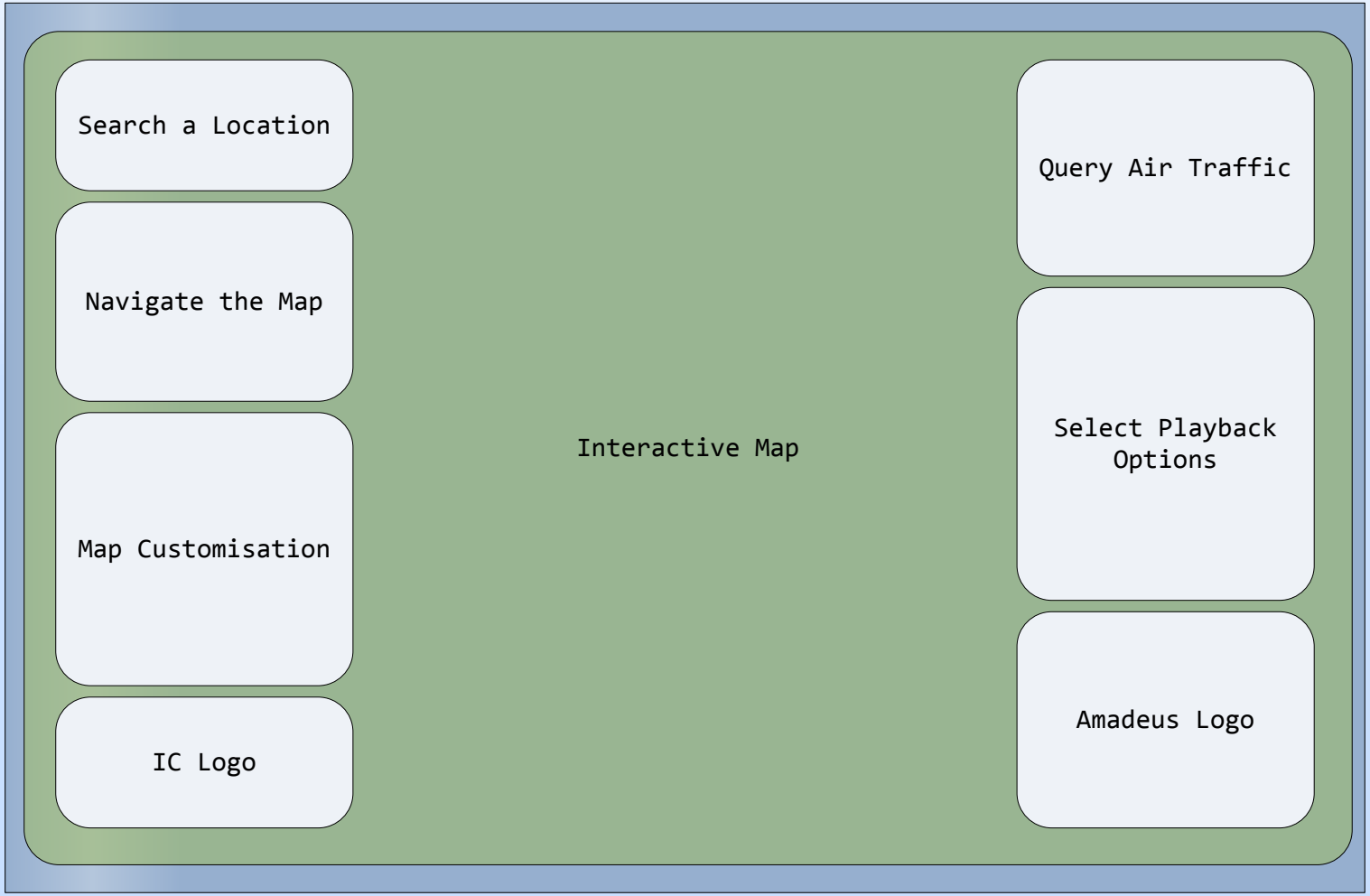
  BRD, 2010/01/22 23:29:39, HKG, LHR, 000414B5A342315B

# Parser Demonstration

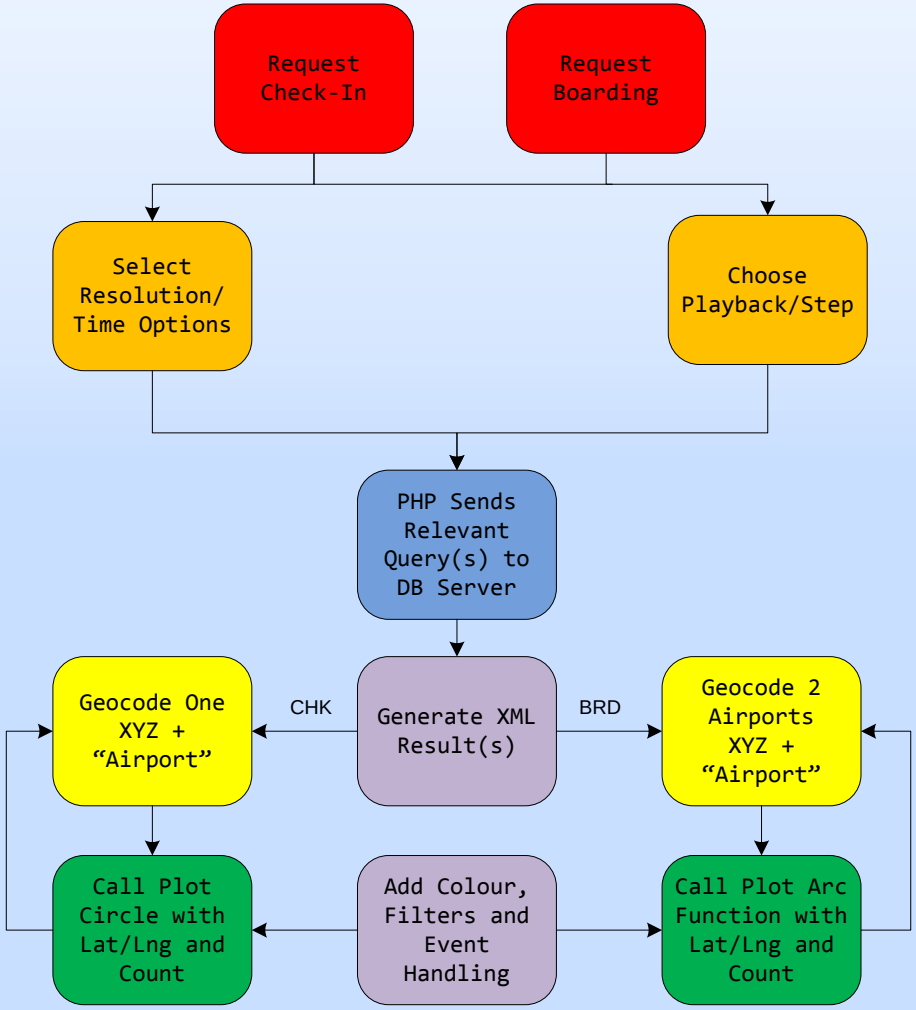
# GUI Functionality

- Google Maps API:
  - Map options + navigation
  - Functions for creating polygon overlays based on lat/lng
  - Geocoding of locations
- Flash operations:
  - All event handling
  - Run back-end PHP scripts for querying departmental database
  - Calculation of polygon geometry for correct display on map
  - Aesthetics – variable opacity, glow filters

# GUI Design


Search a Location
Navigate the Map
Map Customisation
IC Logo
Interactive Map
Query Air Traffic
Select Playback Options
Amadeus Logo

# Query Process



# GUI Demonstration

# Use of Application

- No particular specification given by Amadeus for deployment of app
- Flash is widely used and easily integratable into web-browsers
- Can be ported into a desktop version
- Currently accessed from departmental home directory but can be easily modified to operate on an external server
- Concise application – abstractions provided by Maps API eases the requirement for local storage e.g. just retrieve map tiles

# Achievement of Objectives

- ✓ Able to extract check-in/boarding messages from multiple log files in EDIFACT format – passing relevant information to GUI
- ✓ Implemented a database management system to handle storage and increase communication efficiency with user interface
- ✓ Illustrated check-in and boarding information in a high-quality graphical and textual user interface
- ✓ Enabled a number of features for user-interaction, allowing intuitive and effective querying of airline traffic

# Future Extensions

- Flash – Google update to disable wrap around 180$^{th}$ Meridian
- Joint illustration of check-in and boarding data
- Selection of airport to choose particular data to visualise
- Organisation of database into time segments – rolling system
- Real-time visualisation of airline traffic
  - Foundations are very much in place – need access to latest data
  - Scripts required to retrieve this data on loading application and for intermittent checking

# Questions?