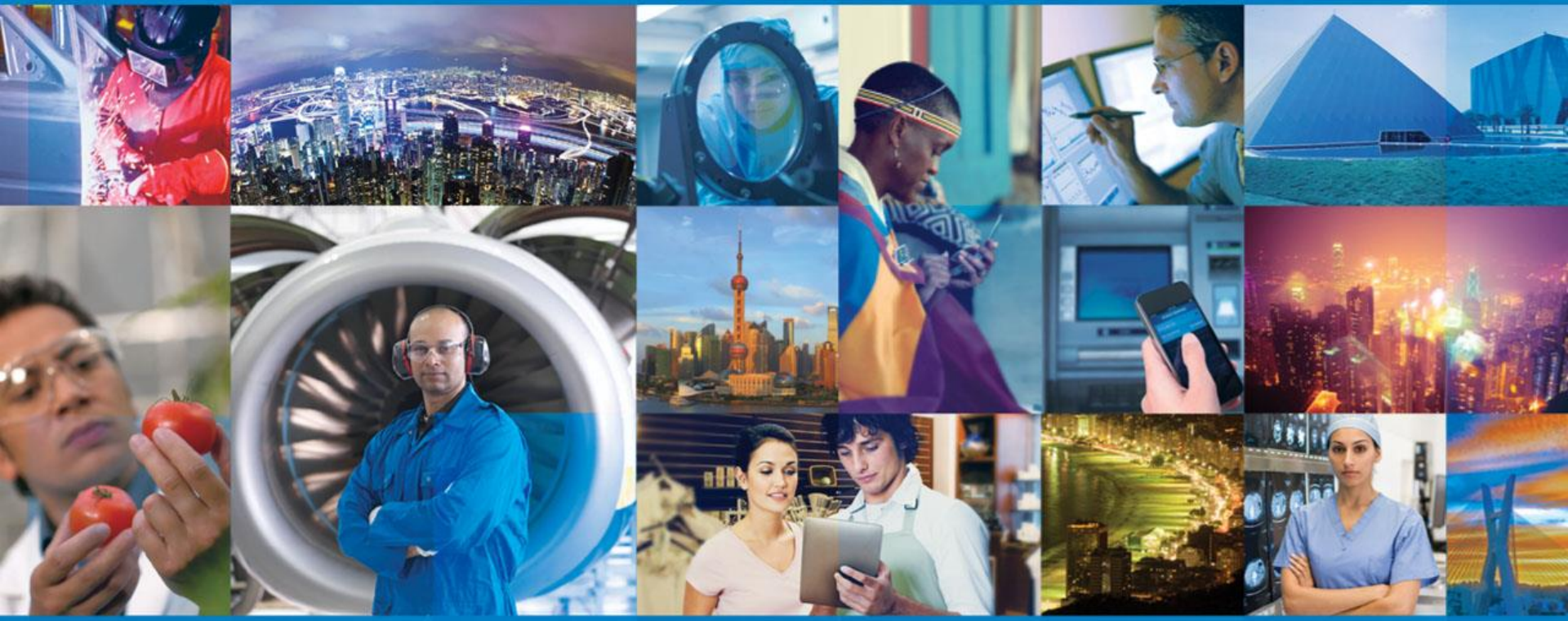
E-Banking fundamentals

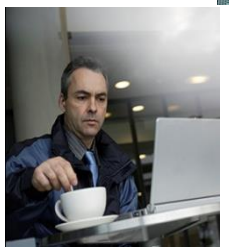


Pushparaj K

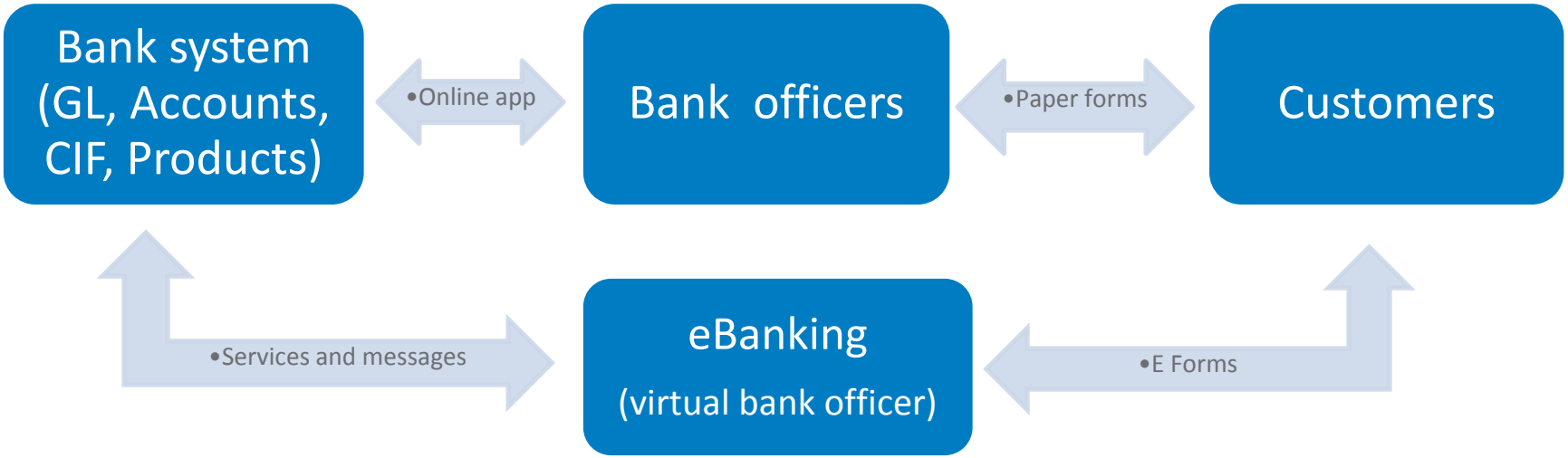
Infosys® | Finacle
Building Tomorrow's Enterprise

Table of Contents

E-Banking fundamentals



Fundamentals – Basic Premise of e-Banking



Fundamentals – Basic Premise of e-Banking

- A Virtual bank user
- Moore effective than a bank officer
- Learn to remember the customer ‘s preferences
- Available 24/7/365
- No holidays for this officer
- Able to control access, manage limits more effectively than a bank user

Responsibilities of Bank system vis-à-vis e-Banking

Bank System	E-Banking
Maintenance of customer data by users employed by bank organization	Recognizing customers and providing self managing feature on own accounts.
• Regulatory information (Central bank) • Product information (Bank organization and strategy) • Alliances (Payment networks, Trade alliances)	Customer to do his own transactions as against requesting a bank staff to do it on behalf of the customer.
Managing access control	Maintaining customer preferences

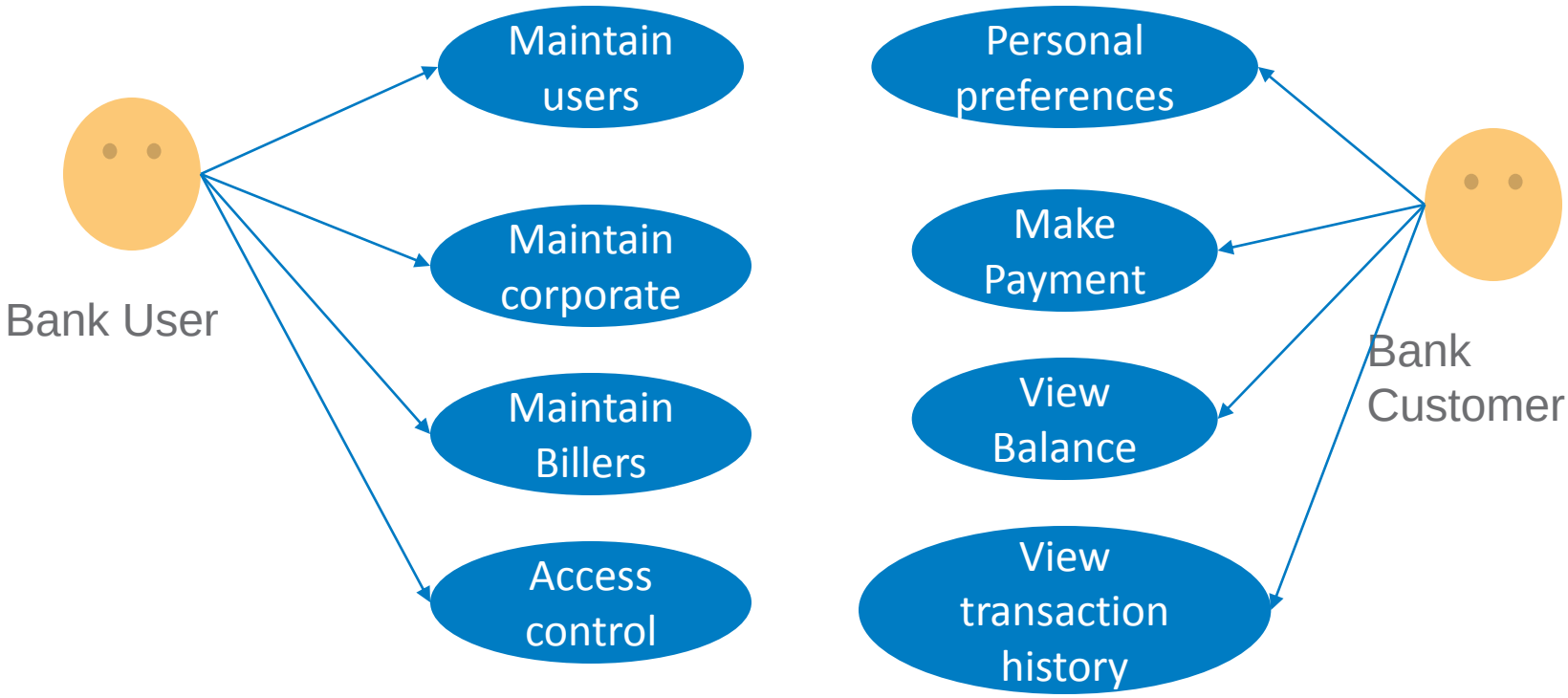
Generic Goals of an Architecture

- Solve generic problems
- Shield application programmers from complexities and changes in OS, App server, database, network
- Make possible a simple and uniform interface to code functionality in the product
- Define problem solving methodology
- In the most refined state, designers and developers will need to work with only the functional requirements of the product

Architecture and use case level problem solving

Architecture level problem solving	Use case level problem solving
Deal with generic issues, applicable to many use cases	Design of functionality, specific to a use case
Create a simpler interface than what exists to help user learn a bit and achieve a lot	Use case developers have to remember to follow each time
Hide generic issues in abstractions	Will have to be tested each time

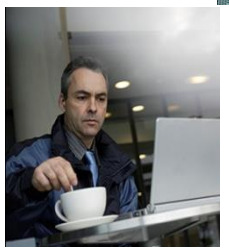
E-Banking basic use cases



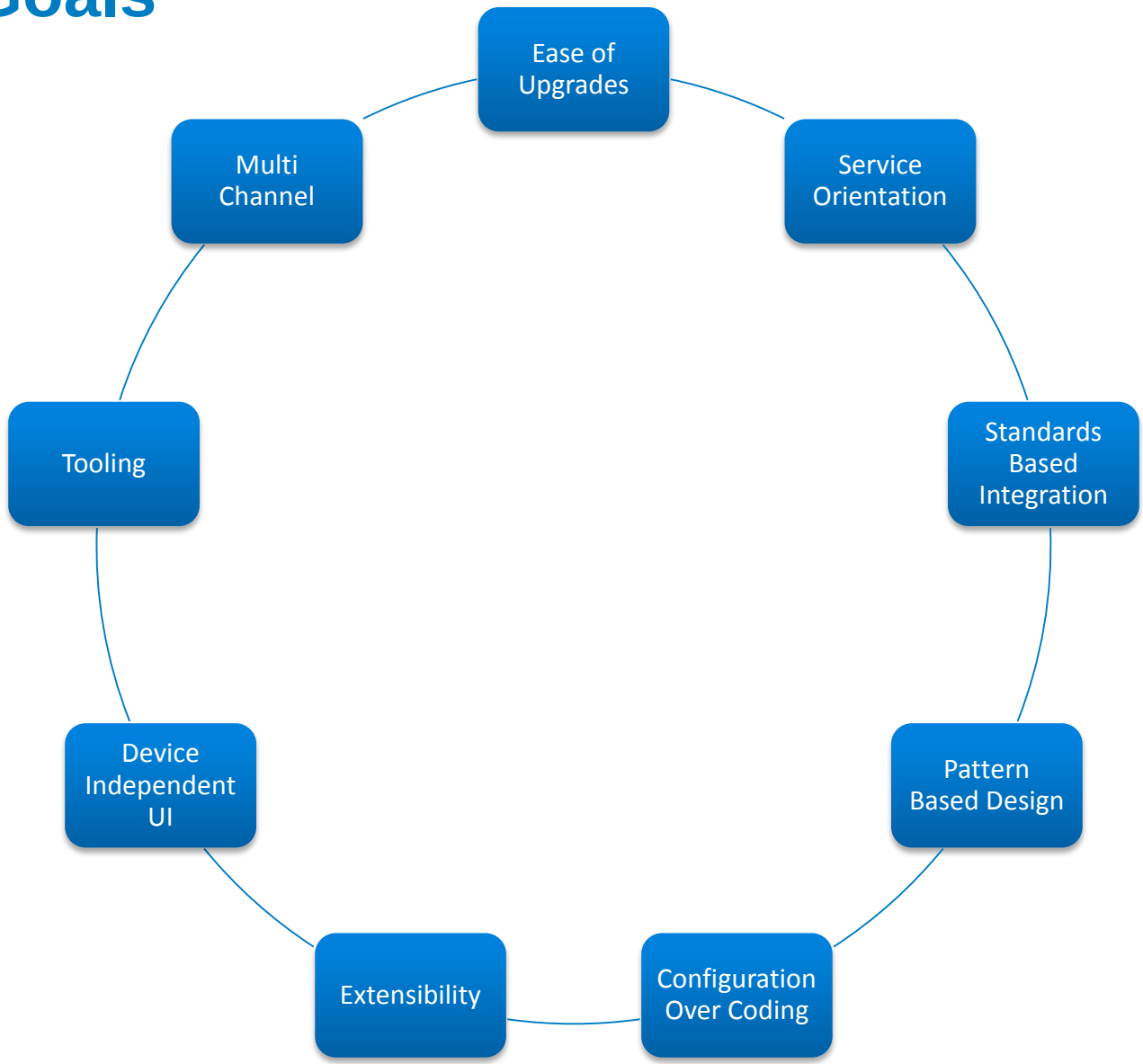
Different Patterns in e-Banking

- User updating preferences (**Maintain internal logical File**)
- User making a bill payment (**Maintain External File**)
- User viewing the list of transactions on an account (**Read External file**, paginated)
- Admin viewing list of users (**Inquire internal file , paginated**)
- Print password in bulk (Maintain Internal File, batch)
- Report of activities of a user (Batch report of Internal file)
- Statement of account sent by automated mailer to user (Batch report on External file)
- Admin inquiring details of a user (**Read Internal File**)

Finacle E-Banking Architecture Overview



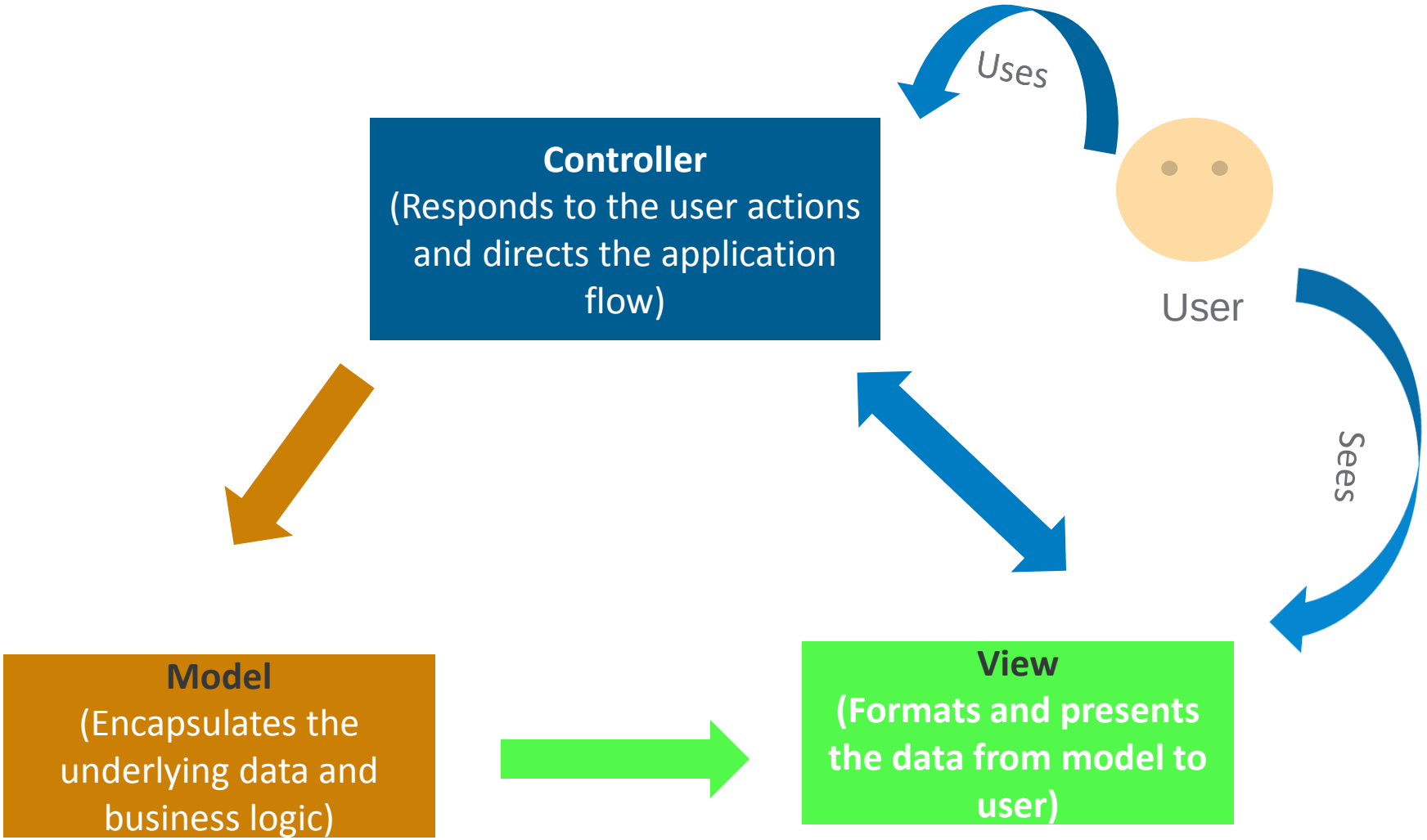
FEBA Goals



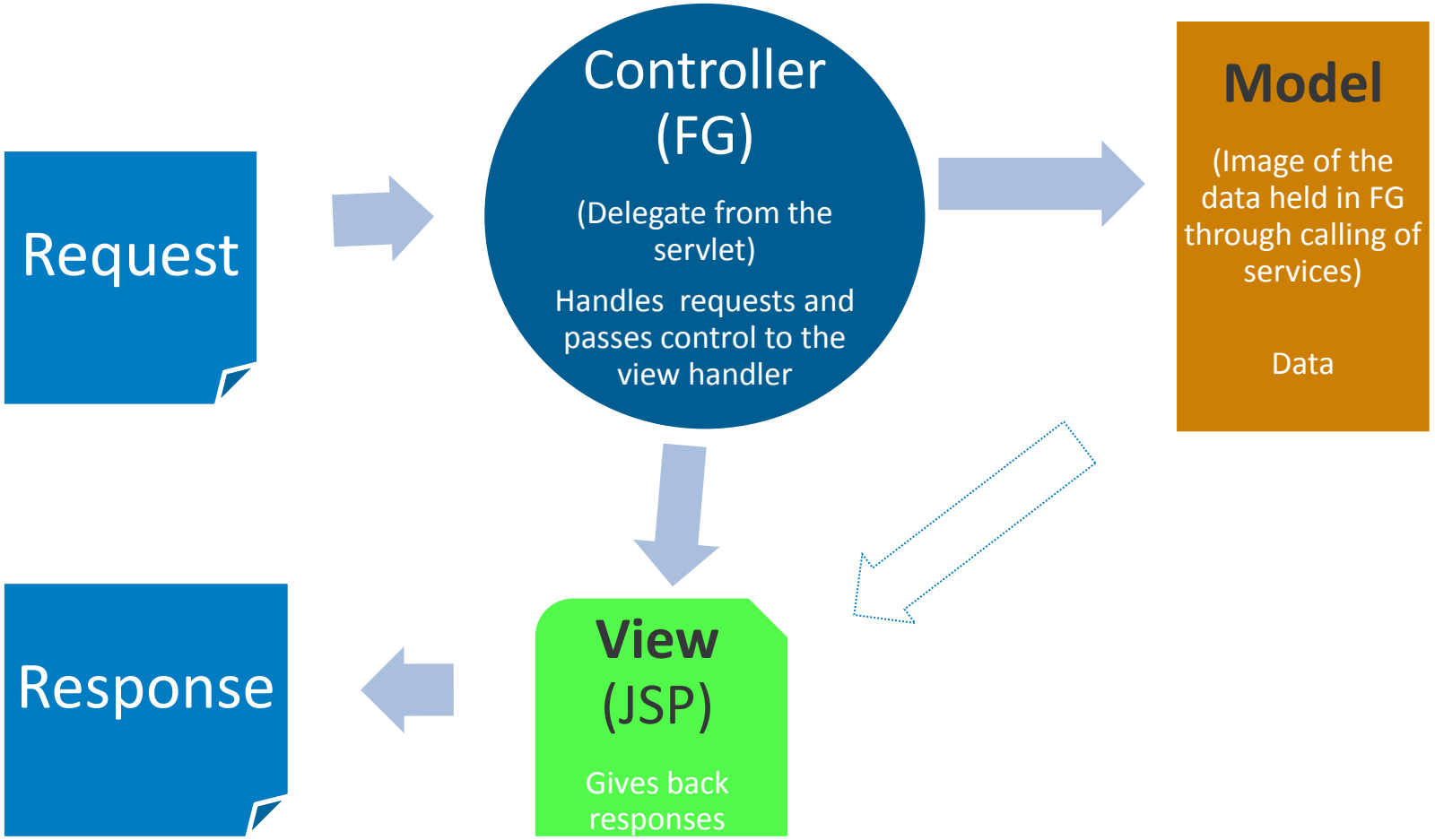
Technical Goals

- **Better Usability**
  - Clear Segregation of UI from Processing Logic
  - Quick Turn Around time for look and feel changes to the site
- **High Development Productivity**
- **High Customizability**
  - Ease of configuring new validations
  - Ease of integration with multiple backend hosts
  - Configuration-driven Integration
- **High Maintainability**
- **High Performance**

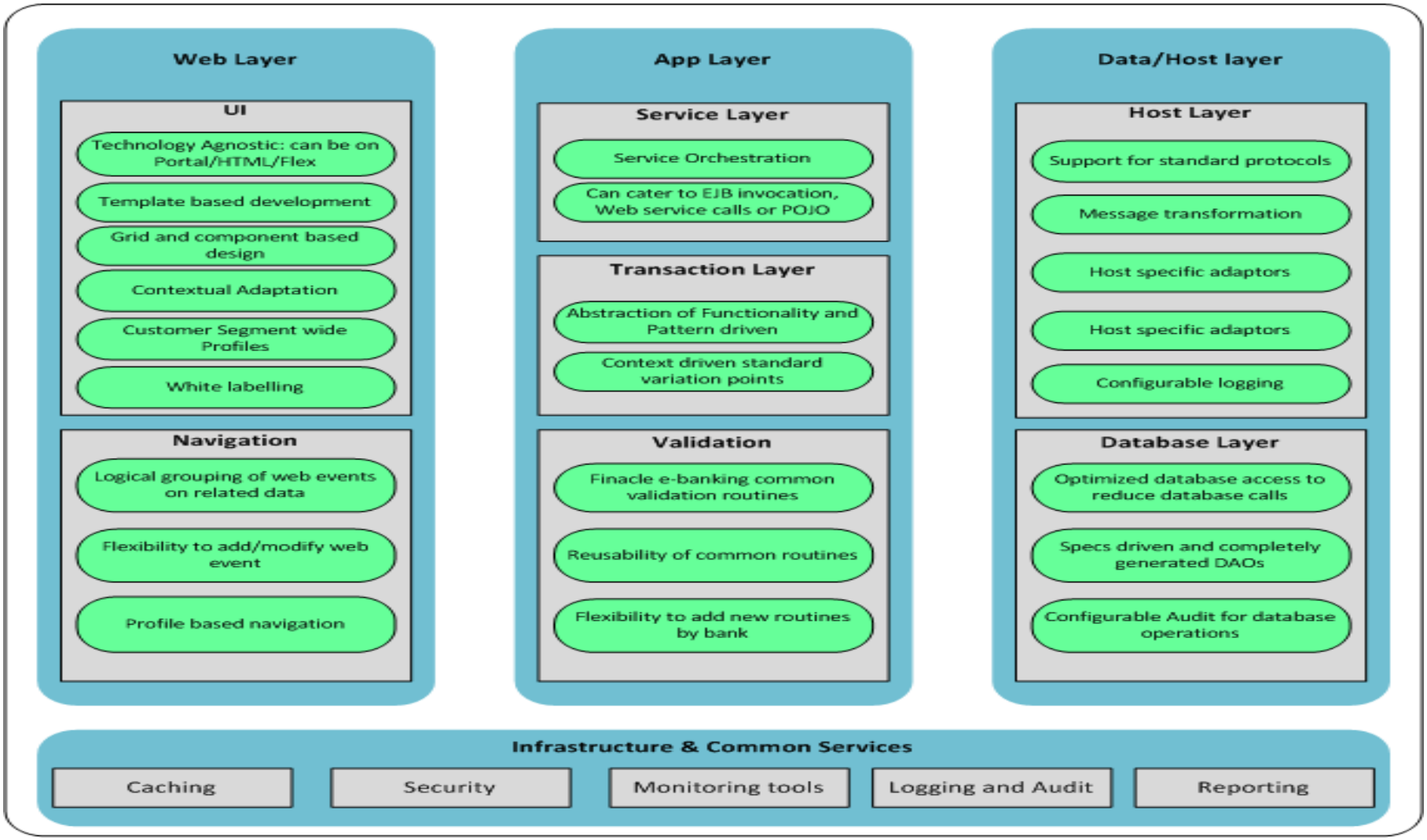
MVC Architecture



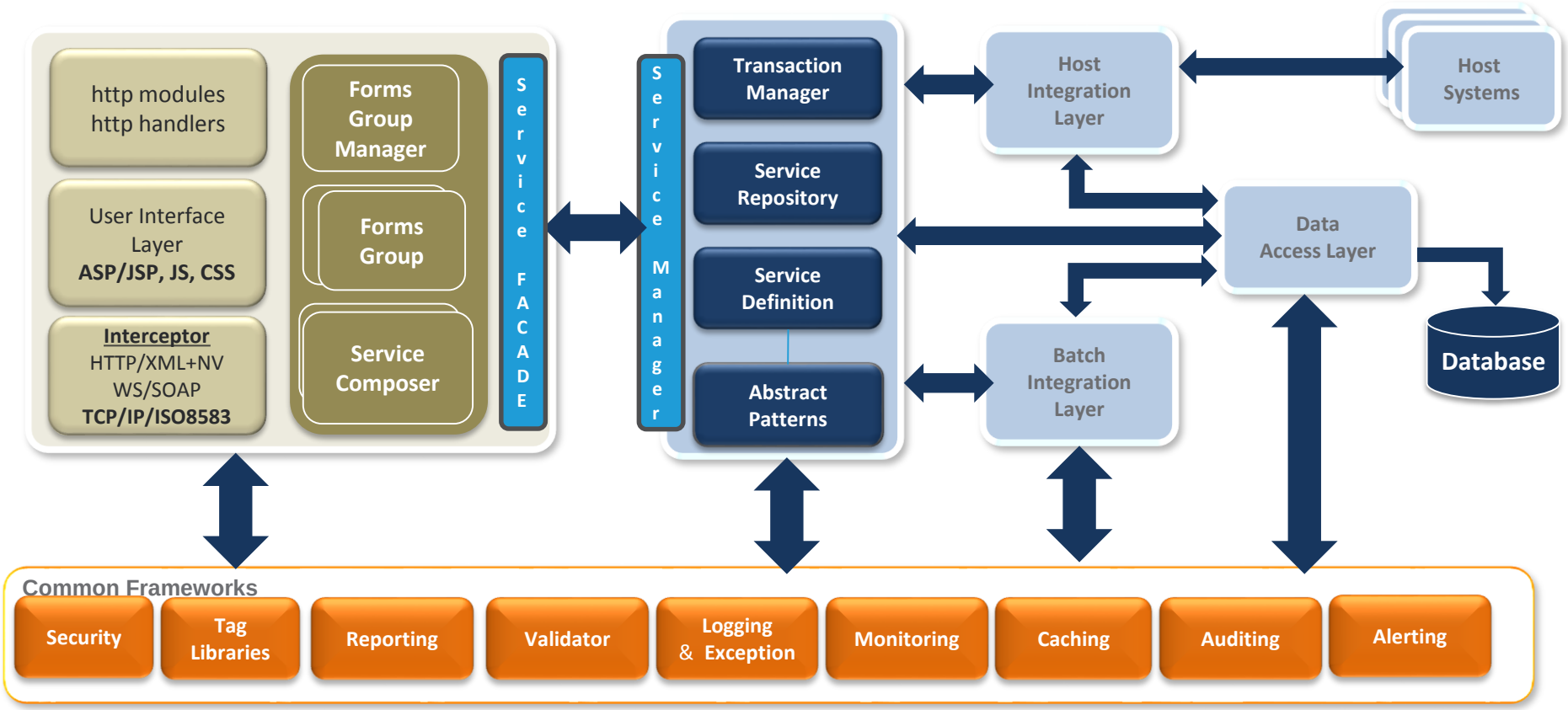
MVC Pattern and Implementations



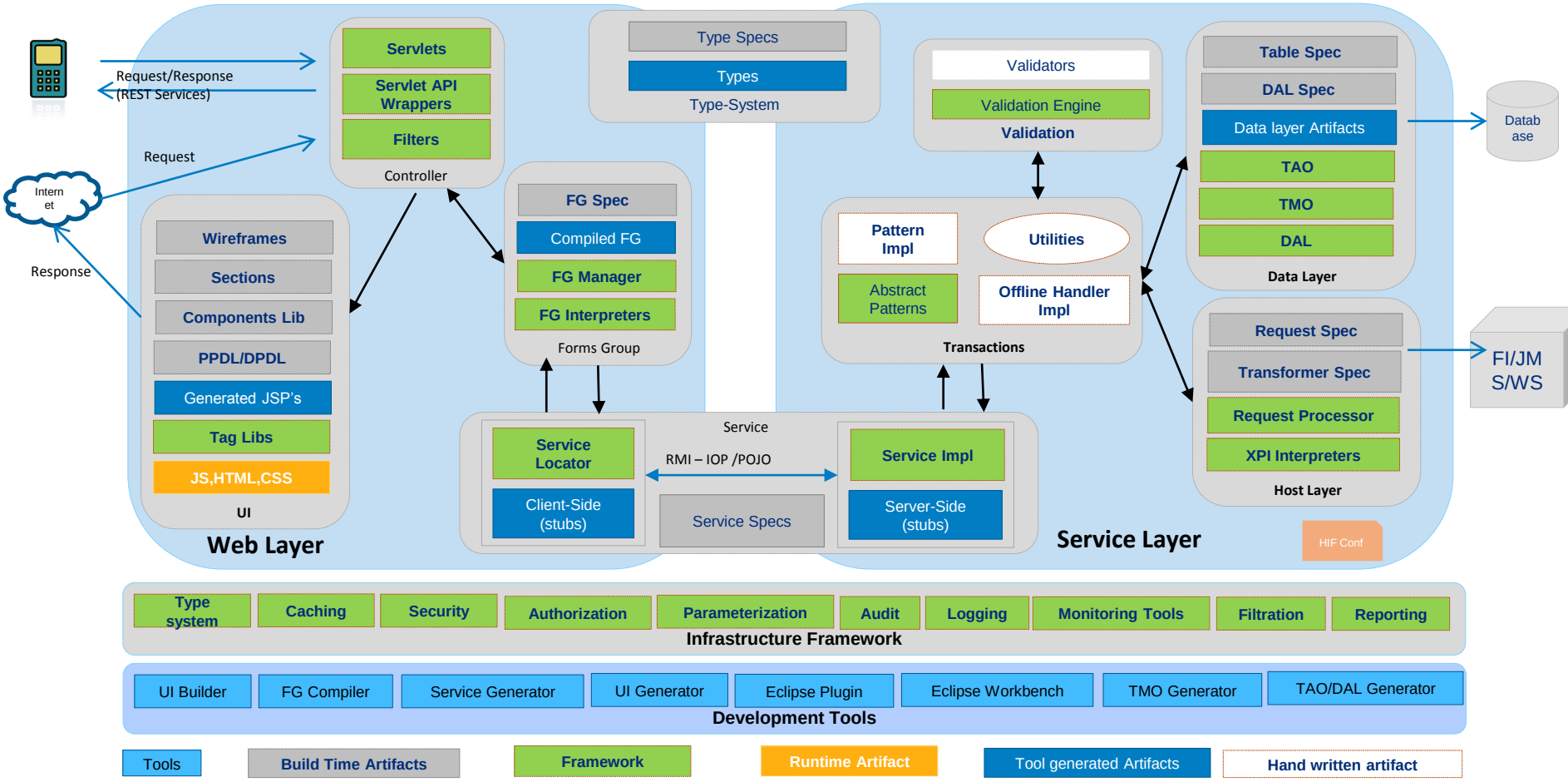
Application Architecture



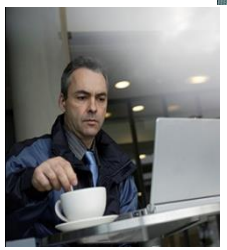
FEBA Architecture



FEBA Architecture

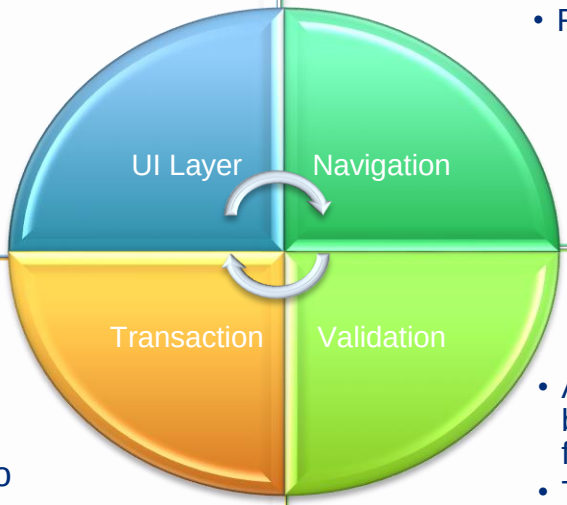


FEBA Framework Overview



FEBA Basic Framework

- Grid Based Approach
- Page content abstracted into well known patterns;
- Tool generated UI
- CSS files driving the look and feel
- Extensive use is made of JSTL tags;
- Works in JS and Non JS
- Relevant WCGA guidelines Compliant
- JAWS 10.0 Compliant



- Grouping of logical set of web actions on a related set of data.
- Drives Navigation
- Specs Driven
- Invoke the Business Service
- Return the view to be displayed

- Pattern driven approach e.g. Local list inquiry pattern
- Abstraction of functionality so that easy to implement.
- Holds the commit Boundary
- Spec Driven

- A configurable set of routines are bundled with Finacle e-Banking for common validations
- The validation framework allows easy extension to manage more complex validations.

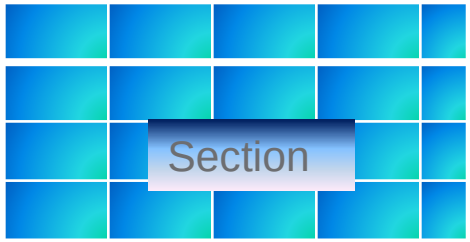
Framework Responsibilities

- **UI Layer**
  - Present data well to the user
  - Clearly handle only presentation logic and nothing else
  - Accelerate development of UI
- **Forms Group**
  - Handle actions from Web
  - Update model
  - Chose the next view
- **Service Layer**
  - Hide Complexities like Database Connections
  - Manage Commit Boundaries
  - Ensure ACID Properties
  - Concentrate only on functional requirements

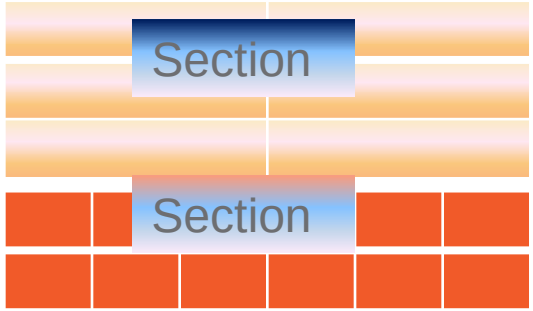
UI Framework

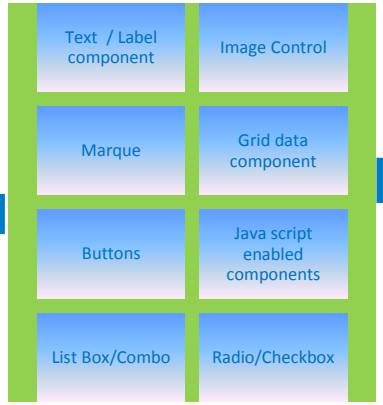
- The UI Framework is based on grids and components.
- A page is based on the Grid pattern and is divided like a graph sheet. Each page consists of a number of grids and each grid contains rows and columns which forms cells.
- Components are the basic building blocks of the UI. Examples of components are textbox, combo box, radio button etc. Components are dropped in the cells of the graph sheet.
- Page Content has been abstracted into well known patterns.
- The whole framework and the components are built as Tag Libraries and there will not be any use of HTML or scriplets (java code) inside a JSP.
- The Look and Feel of a UI can be easily altered. CSS files drives the look and feel of the page.
- Works in JS and Non JS mode.

GRID based User Interface

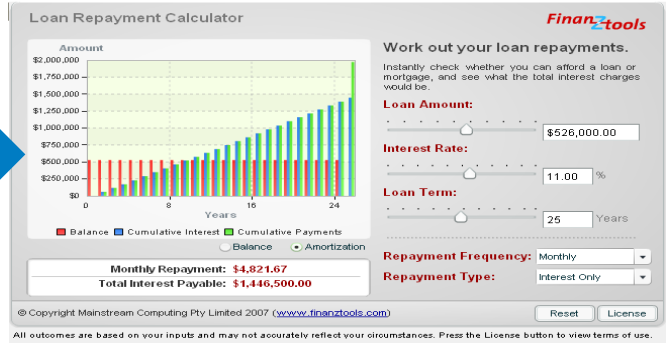


Wireframes

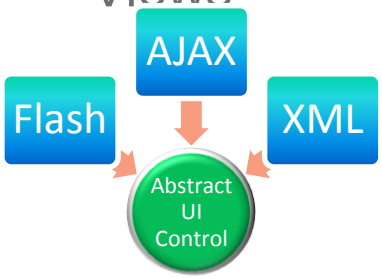




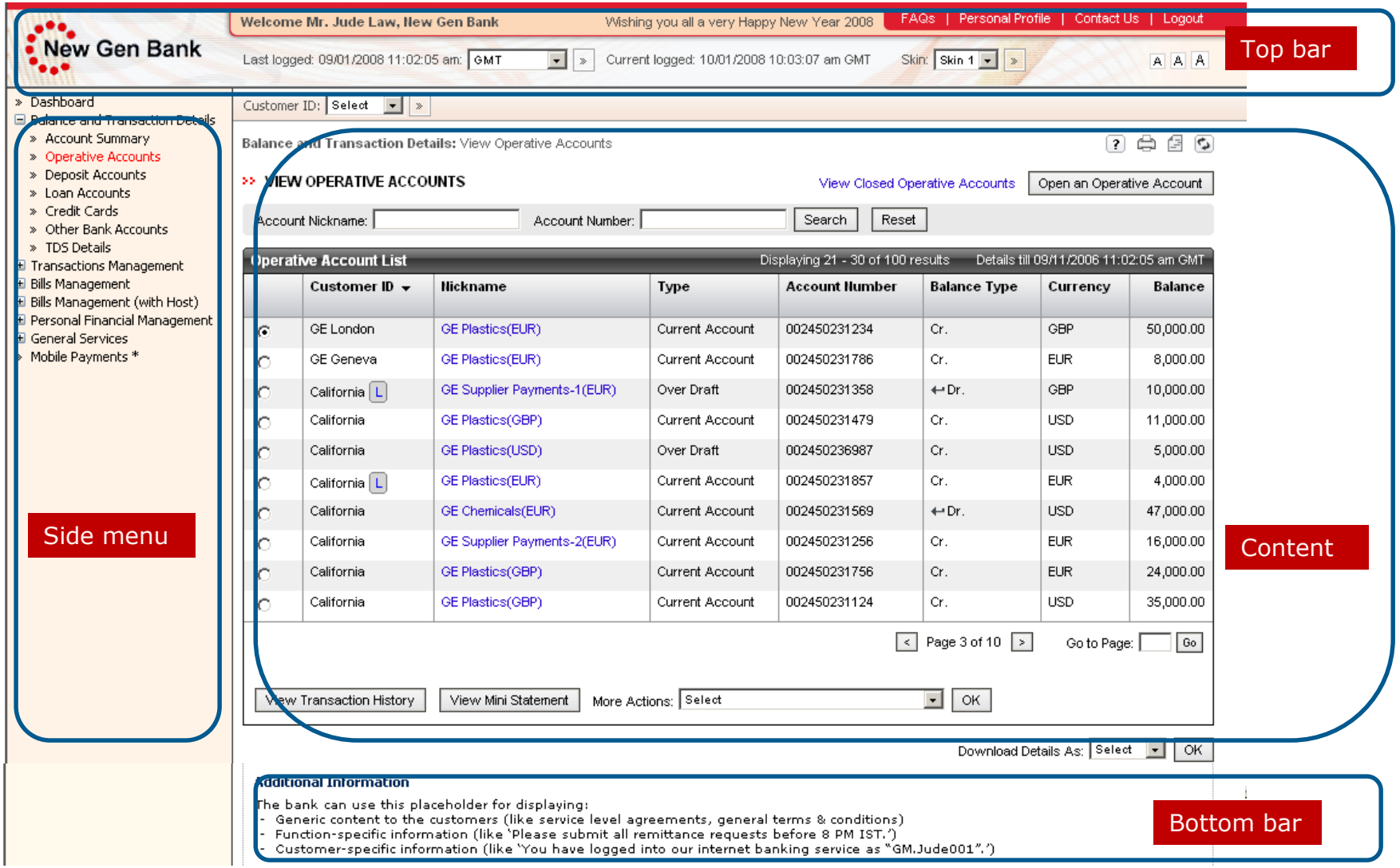
Components



Views



UI Framework



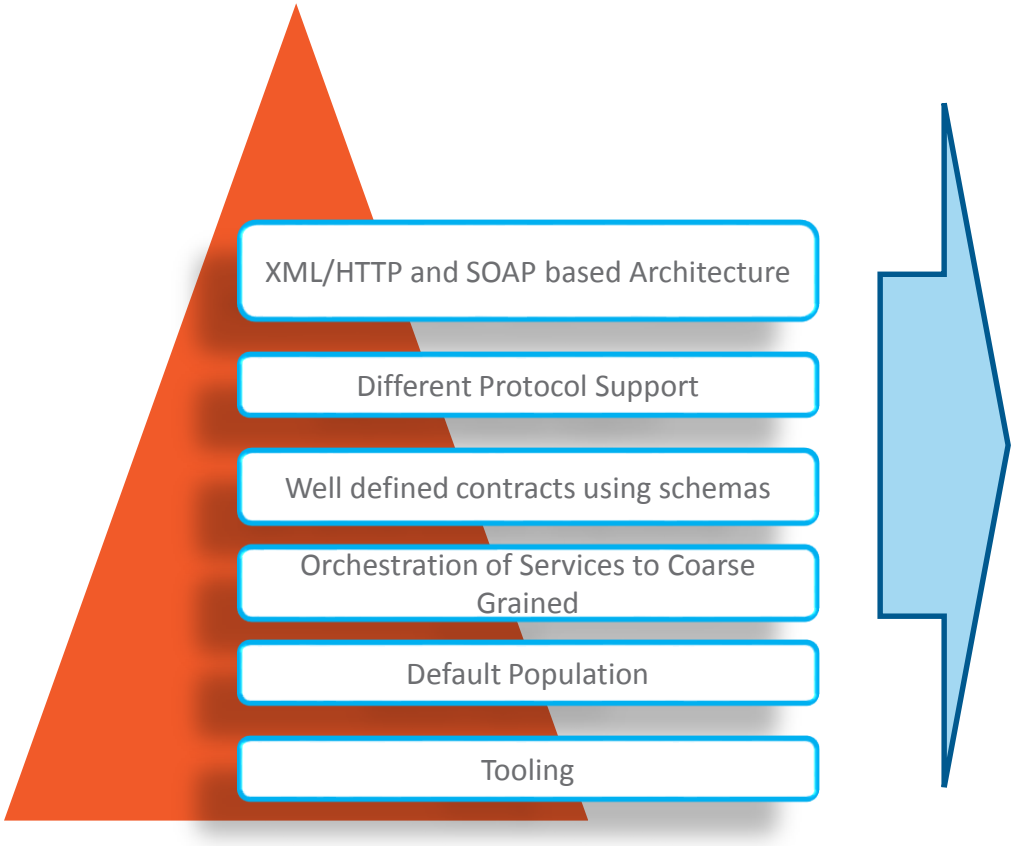
Forms Group Framework

- Also referred to as “FG”.
- FG is a logical set of web actions on a related set of data.
- FG is a component which handles a set / group of related user actions initiated in a UI.
- Drives Navigation - FG is the mechanism through which one defines Page Navigation and transaction data management for a Use case in the web layer.
- Specs Driven - FG specification has been captured in a XML specification file called FG spec. Event processing logic and data are encapsulated within this specification file.
- Invokes the Business Service
- Return the view to be displayed.

Transaction Framework

- Business tier of the e-Banking system
- Transaction layer comprises of the actual business logic written by developers to accomplish the requirements
- Invoked through the service layer
- Patterns identified across use cases of e-Banking
  - Read from Local Database – does not commit
  - Write into Local Database – updates data and commits once
  - Paginated Read from Local Database
  - Read From Host
  - Update Host
- Common components abstracted out such that individual implementation classes provide the business logic only
  - Pagination handling
  - Connection fetch/ close
  - Commit boundary

Service Orientation and Open Standards



- Services are exposed as SOAP and XML/HTTP based pattern and can be contacted from a client or through an ESB or BPEL Engine

- JMS/TCPIP will be supported in future.

- The Services are exposed as standard and well defined interfaces.
- Can be called from any third party application (IVR, mobile etc)

- Optimized Services Granularity
- Fine Grained services can be orchestrated to course grained.

- Services can be configured to default populate values based on the channel and the protocol.

- Tools to generate the part of configuration and navigation based on the business specs.

Data Access Layer

- TAO – To handle single record and primary key based DML requirements
- TMO – To handle single record and primary key based file maintenance based DML on a table
- DAL – Used for selects based on non key columns
- Takes care of time stamping (time stamp fields are known as cookie)
- Provides facility to map table fields to value objects
- Caches selected records internally to provide for read consistency
- Takes care of Audit
- Specs Driven
- Completely Generated Code
- List Queries can be altered and code Generated

Host Interface Layer

- Ensure Logically complete Read / Update data from / to EIF as required by any transaction
- Easy configuration to take care of disparate hosts
- Various protocols for interaction with different host system
- Offline data handling
- Transformer layer to take care of formatting of data between e-Banking and host formats and error handling
- Message adaptor and request adaptor components to take care of protocol and orchestrating requirements
- Configurable XML for coding of both request and transformer layer (XML specification similar to FGI)
- Compatibility with multiple Finacle UBS versions

Reporting Frameworks

- A Report is a flat file of formatted data generated from anywhere in E-Banking.
- Reporting framework - framework to enable any kind of report to be generated using the Jasper Reporting Engine.

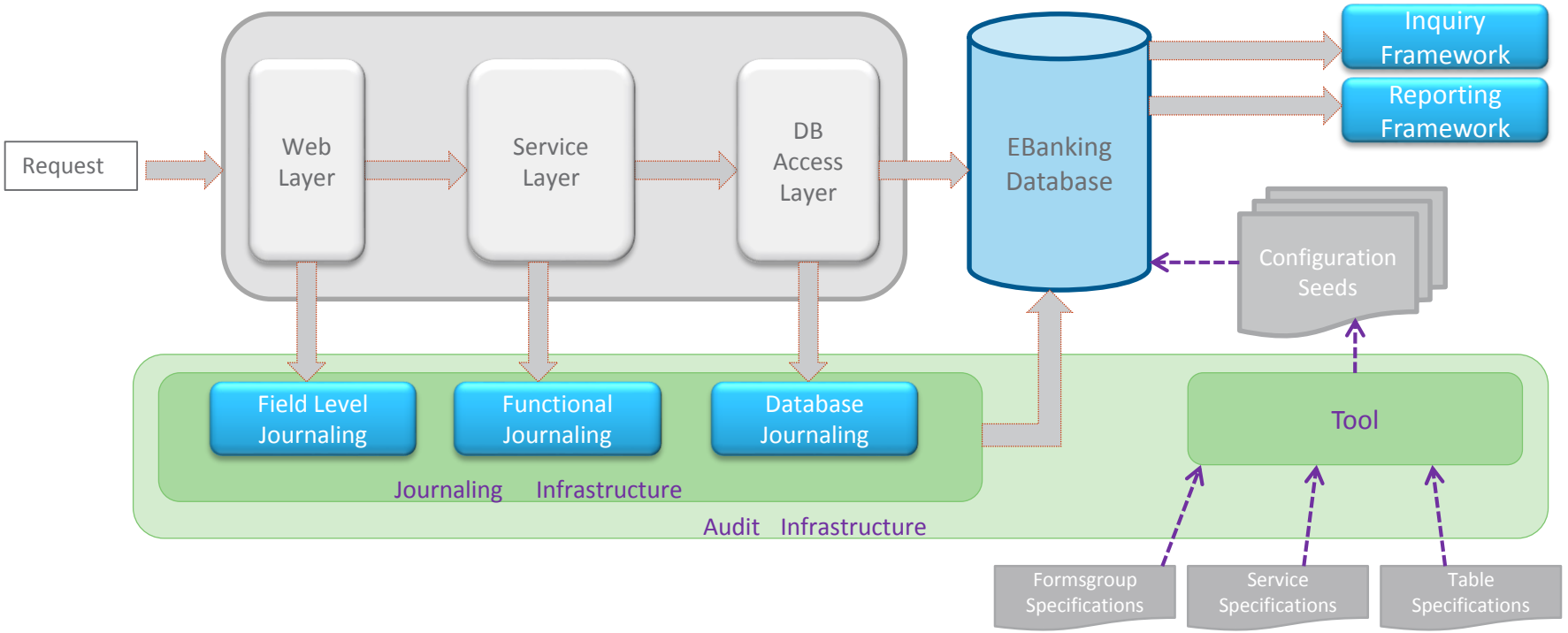
Features

- Supports PDF/CSV/XLS/RTF/TXT/XML/HTML/DAT formats.
- Uniform interface for all types of reports.
- Dynamic report generation from UI without a template.

Logging Frameworks

- Extensive logging and trace framework
- Supports different logging levels
- Easy to debug. Filter based debug. Configuration driven
  - User based filter (trace will be enabled only for configured user)
  - User Type filter (Retail, corporate , Admin , RM)
  - Flow based (funds transfer etc)

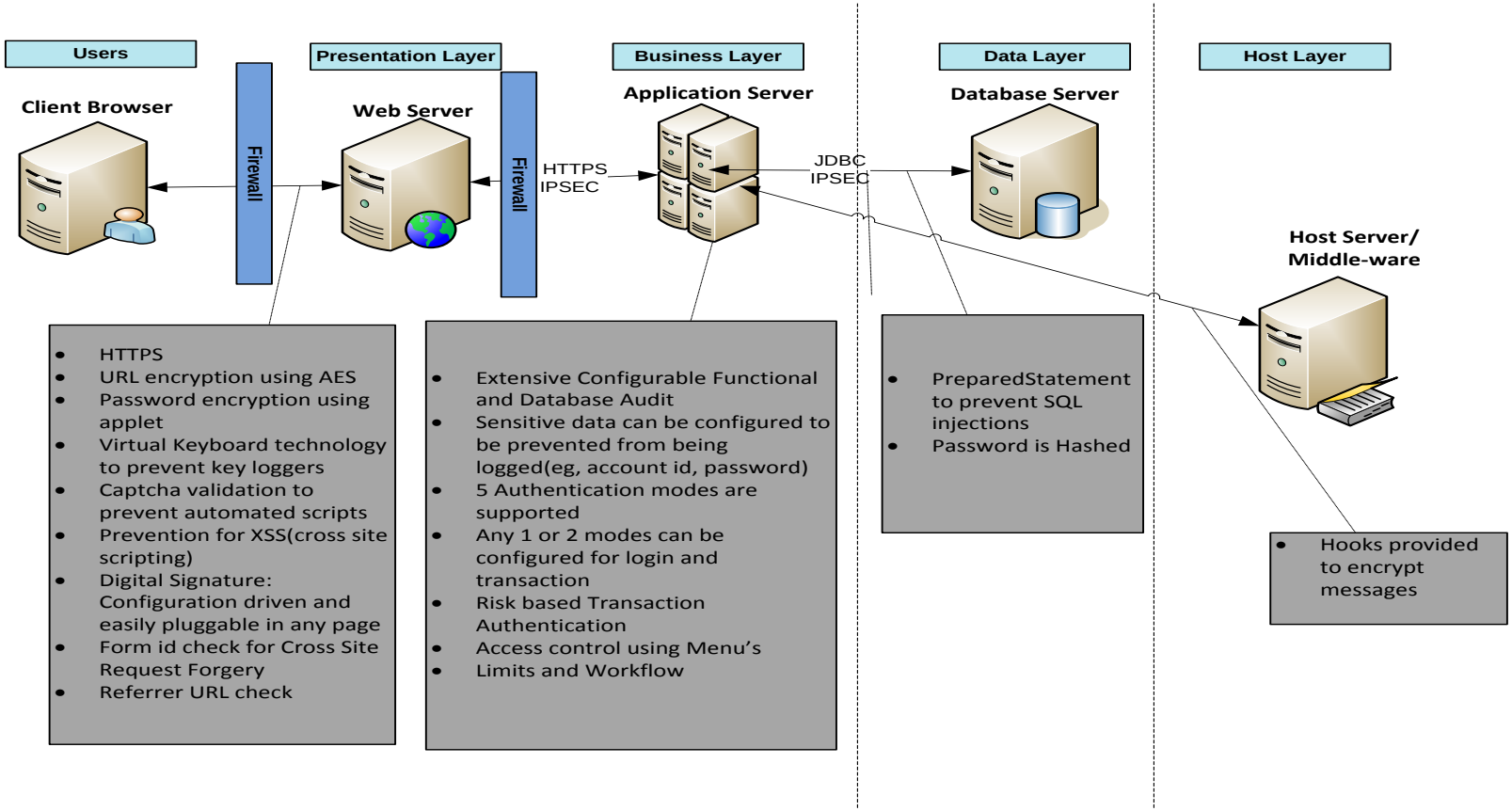
Audit Infrastructure



Caching Framework

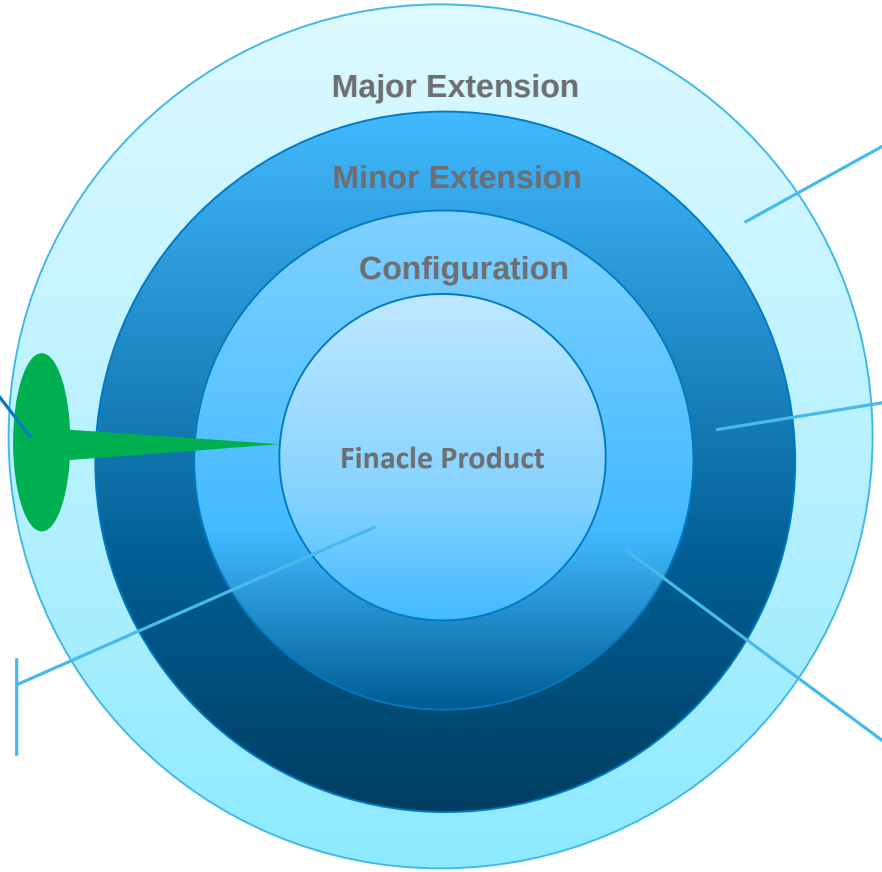
- **Application Level**
  - Information such as error codes\messages and master reference information is cached at this level;
  - It is lazy loaded as required; and
  - Is refreshed at scheduled intervals automatically;
- **User Session Level**
  - *Persistent information*, which will be there for the full lifetime of the session level and is cleared when the user session ends;
    - It consists of information such as User Data, Accounts Information, Access Permissions, Balances and Transactions.
  - *Transient information* whose scope can vary from a single request to a series of requests. After the completion of these series of requests the transient information is flushed from the session cache;
  - User session data is not updated during a user session other than in response to a user action. For example a user makes a payment and the balance for a particular account changes in which case the cached value for the account is updated.

Data Security



Finacle Extension v/s Product

- Extensions in Finacle can be classified into three types….



- Extending the data model;
- Overriding business rules;
- Technical activity requiring detailed Finacle knowledge.

Testing and Maintenance costs increase only with complexity of extension.

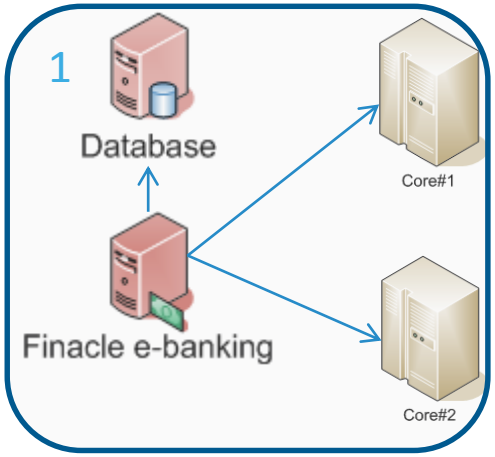
- Modification of existing reports;
- Creation of new reports;
- Integration;
- Adding new file formats;
- Creating new screens or modifying existing layouts;
- Integrating host
- Technical activity requiring basic knowledge of Finacle.

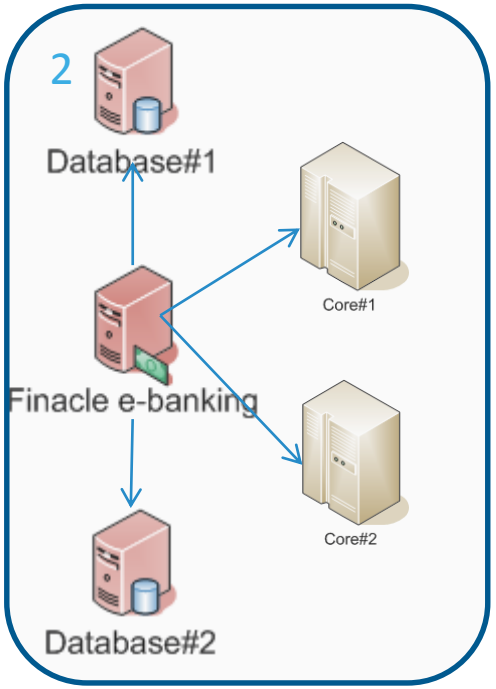
- Never modified as part of an implementation.
- Managed by Finacle.

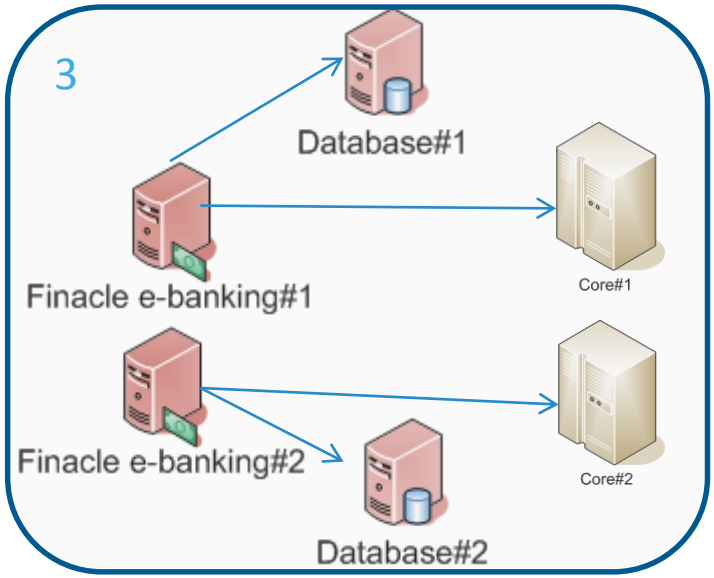
- Master Reference Data;
- Menu & Navigation Profiles;
- Validation;
- Parameterization
- Style sheet Updates.

ME Deployment Options

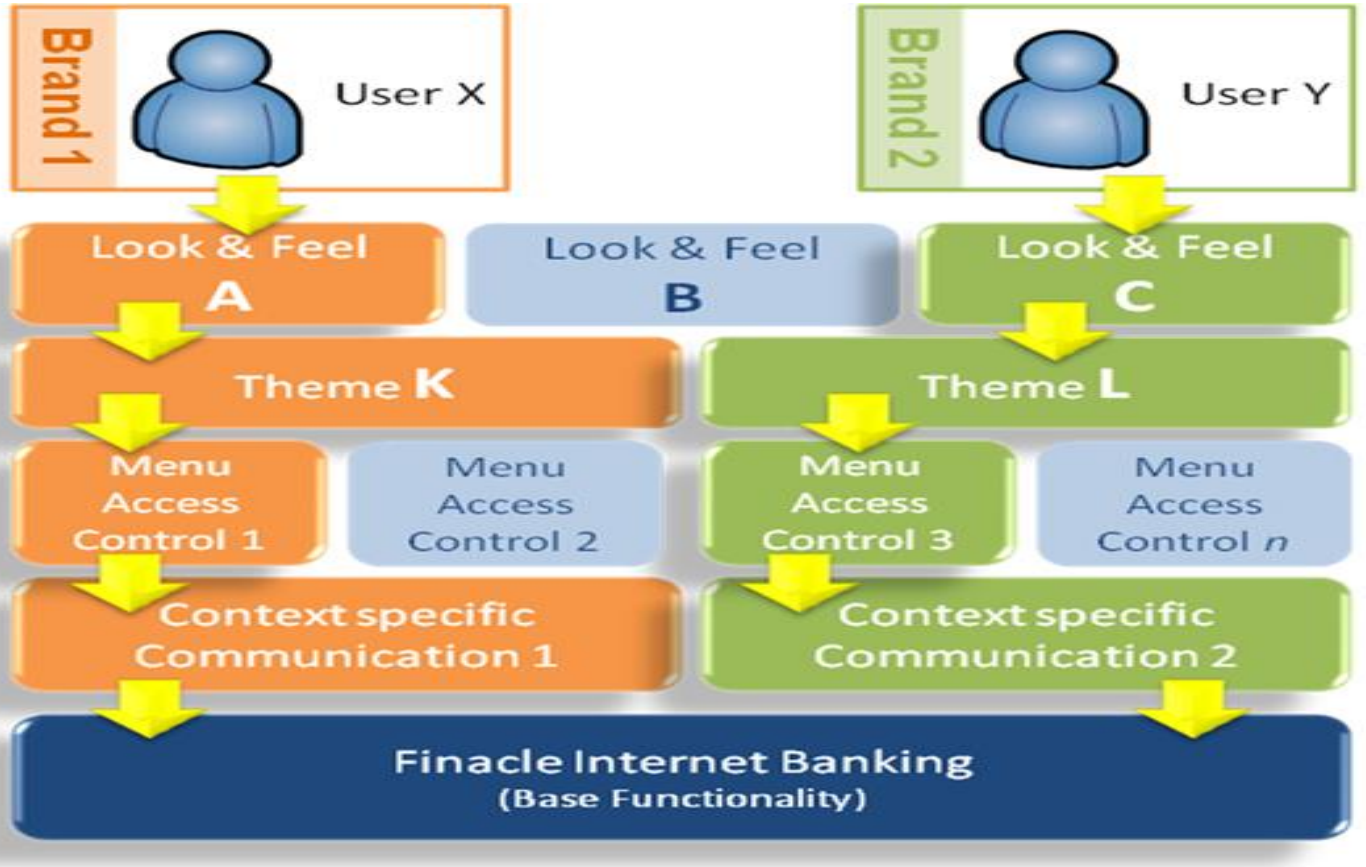
- Flexible Deployment options to suit business needs and structure of Banks.
- Supports deployment of multiple entities in the same installation, with varying degrees of separation and common use.
- Different entities of the bank can have either dedicated or shared resources.







Multi-Branding



Thank You

Infosys® | Finacle
Building Tomorrow's Enterprise