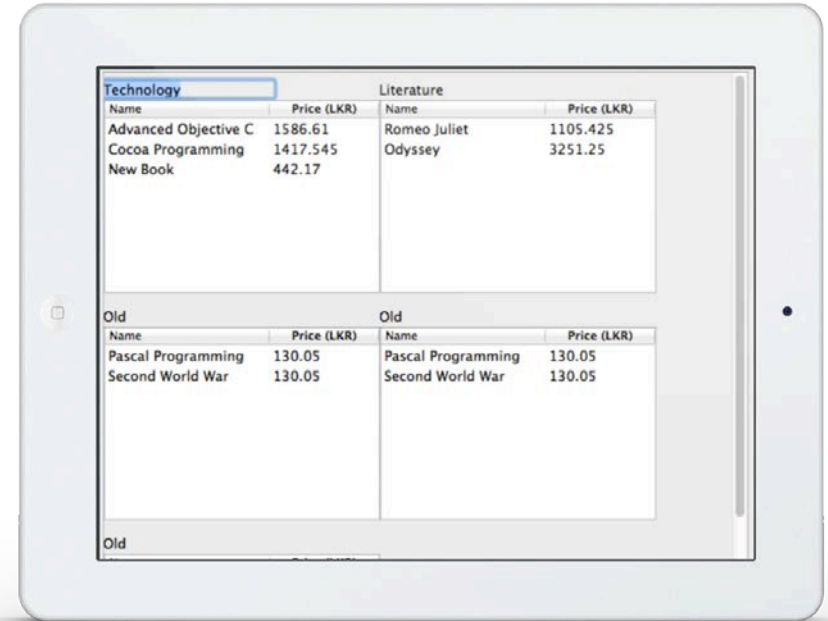


Adapting View Models as a Means For Sharing User Interface Code Between OS X and iOS



Dileepa Jayathilake

99X Technology

SATURN 2013

OUTLINE

Conclusion

Discussion

Implementation

Solution Overview

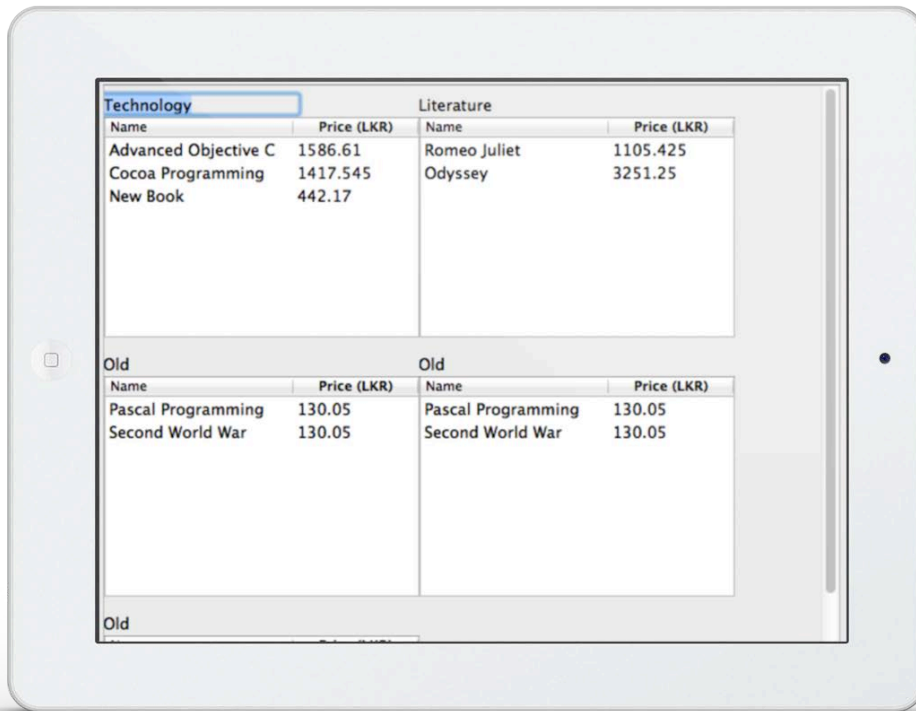
Problem Identification

Background



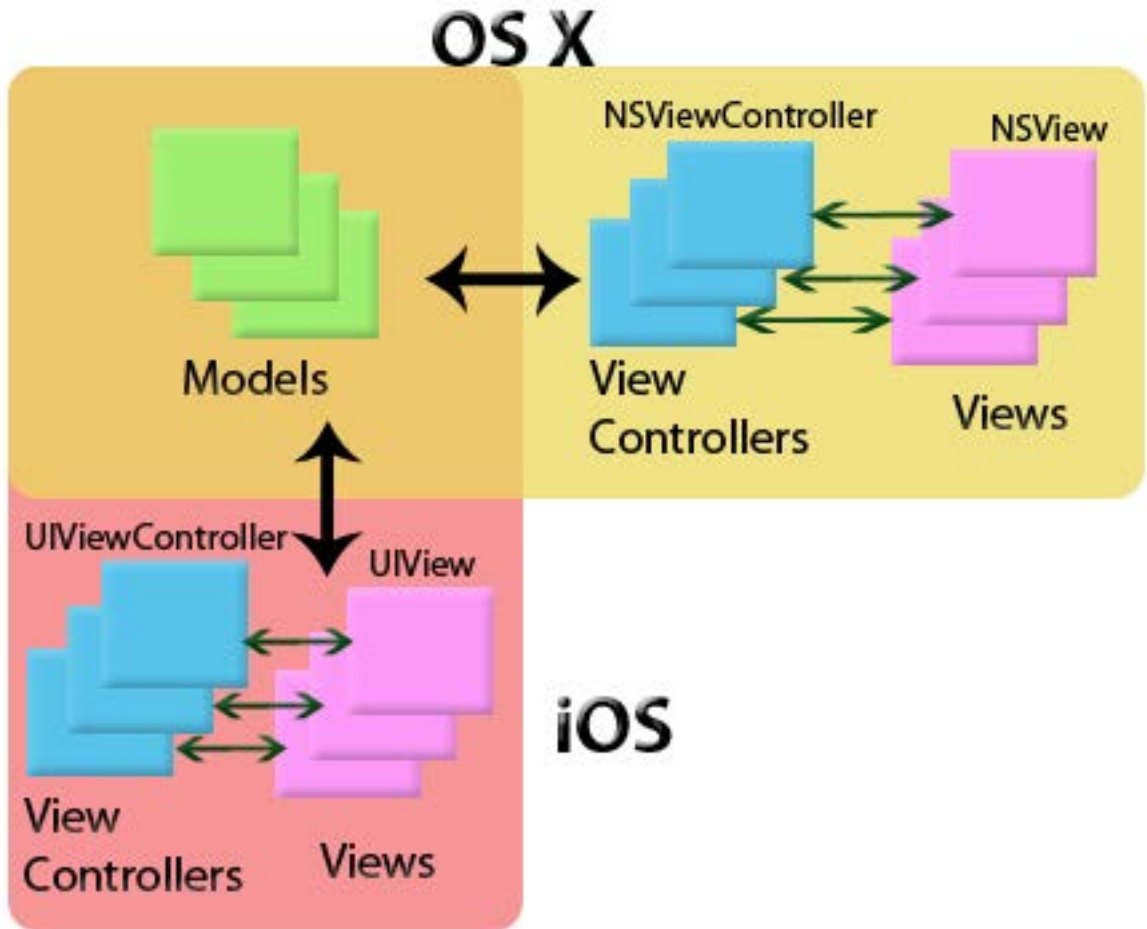
With a significant part in common

Product for both OS X and iOS



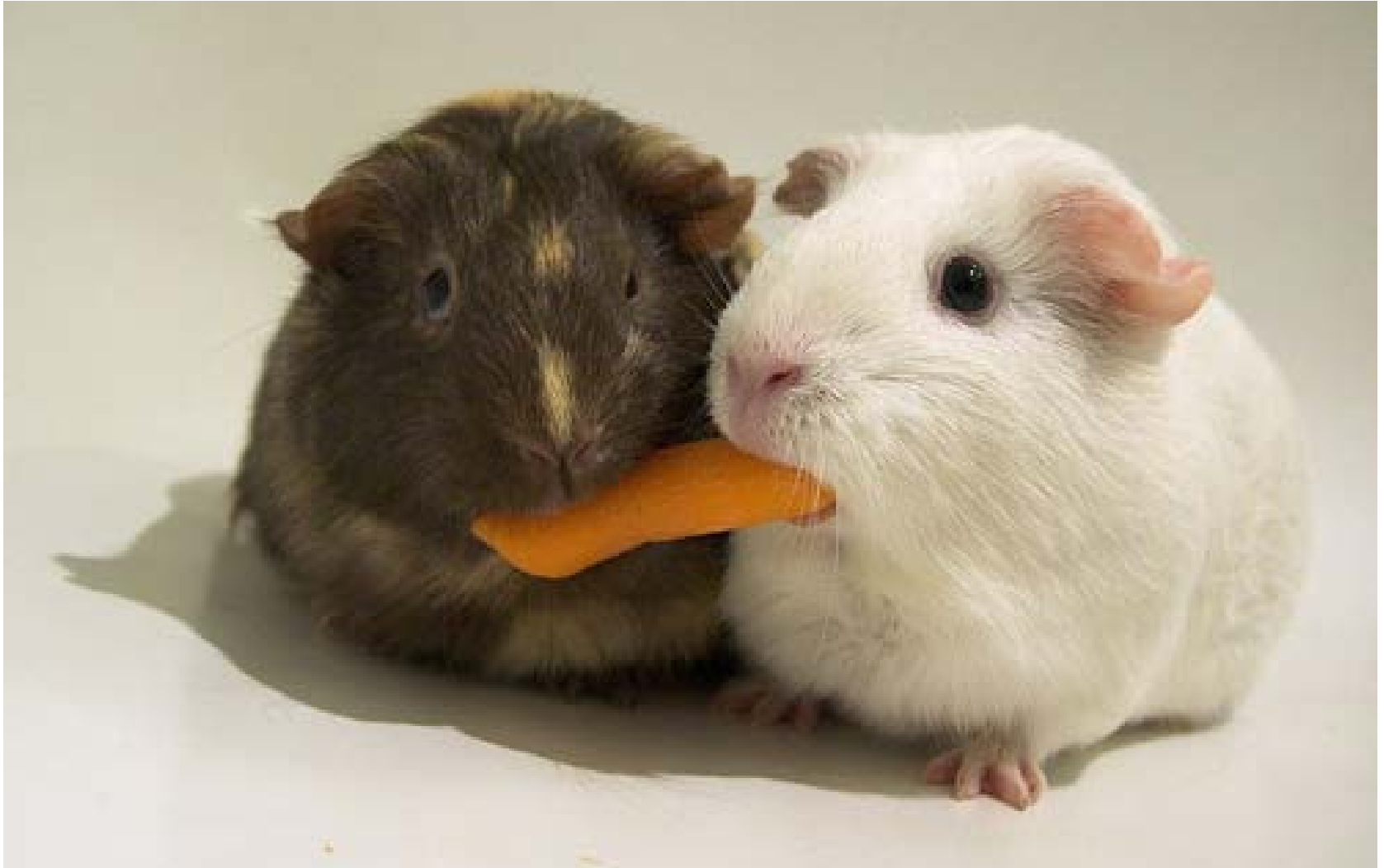
How to reuse code maximally?

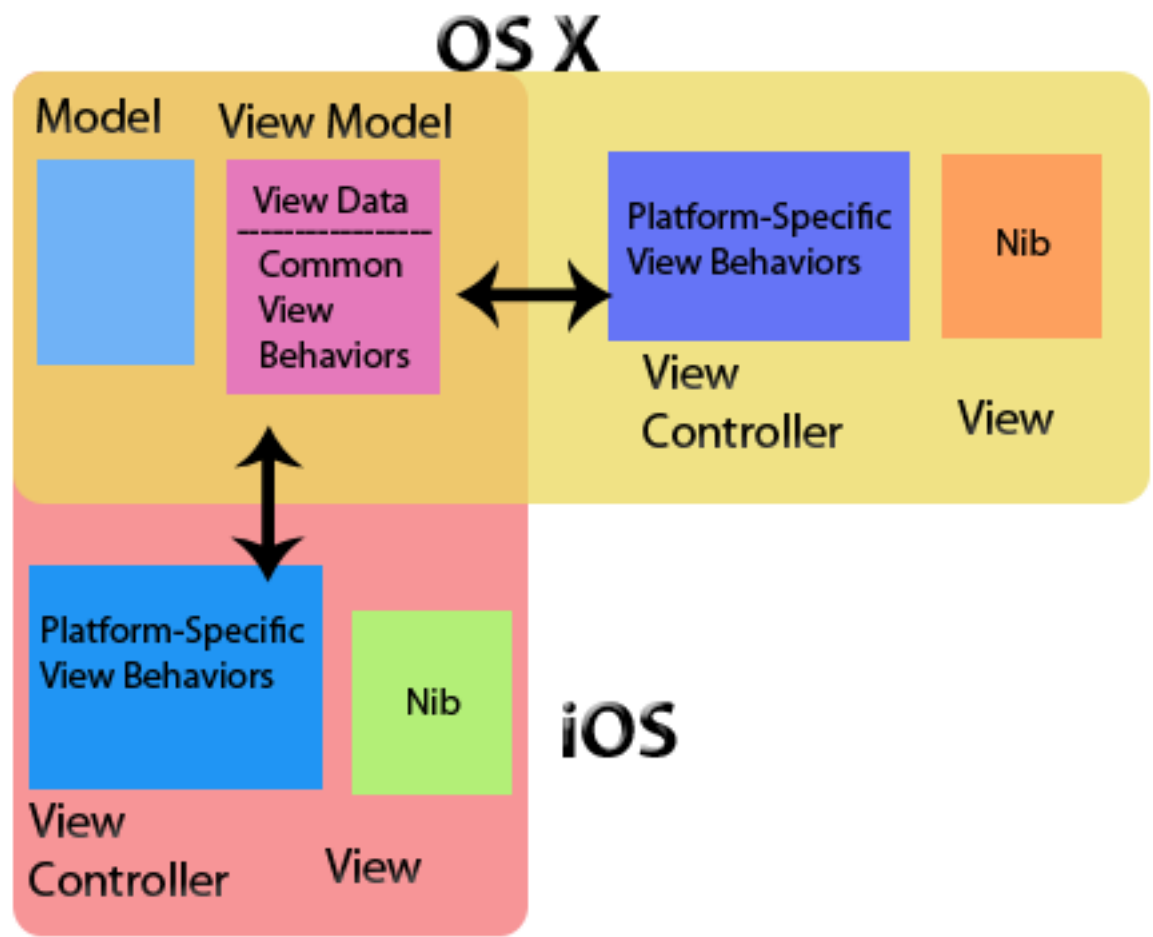
BACKGROUND



**Is it possible to
share UI code?**

**PROBLEM
IDENTIFICATION**







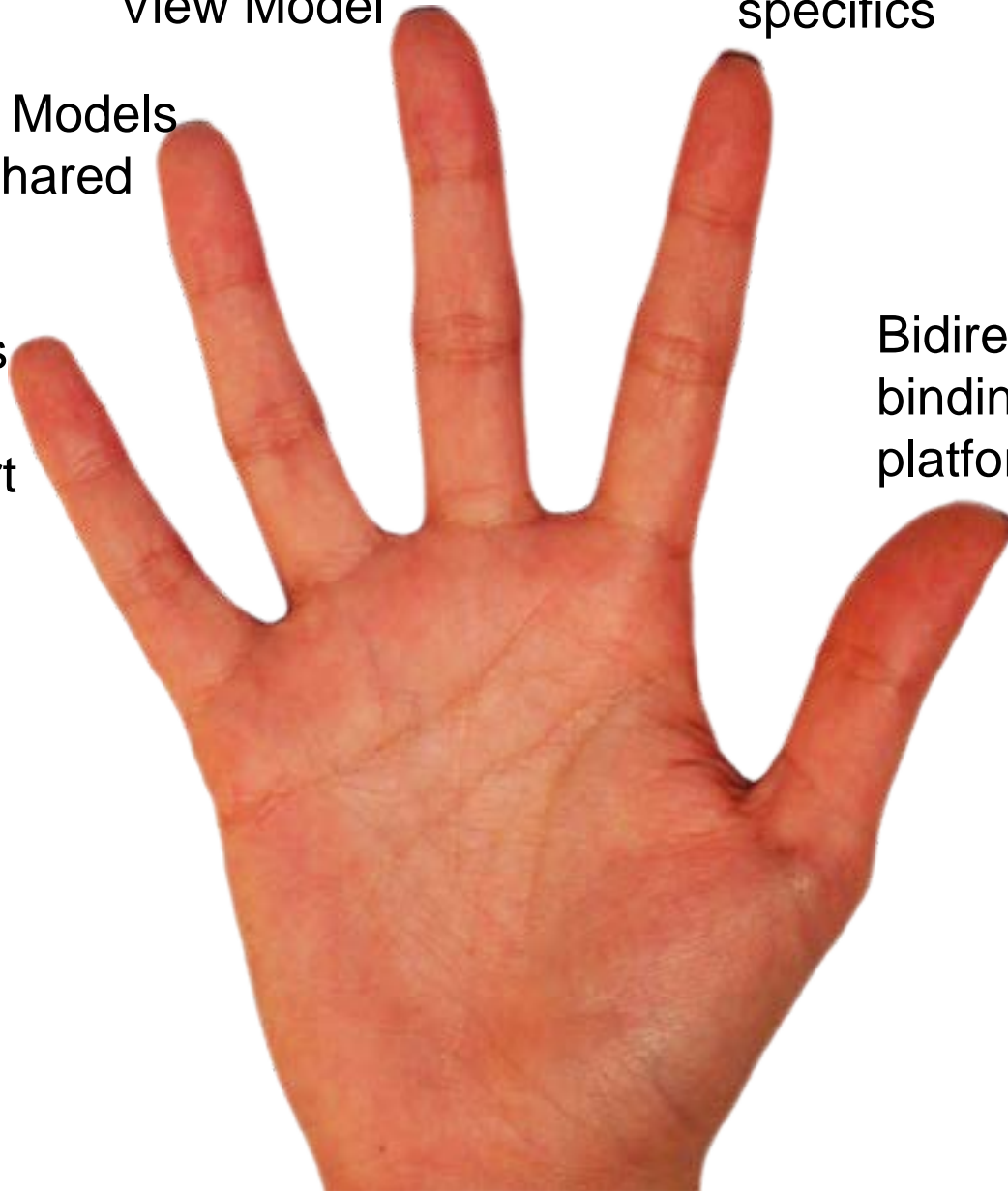
Controllers
hold platform
specifics

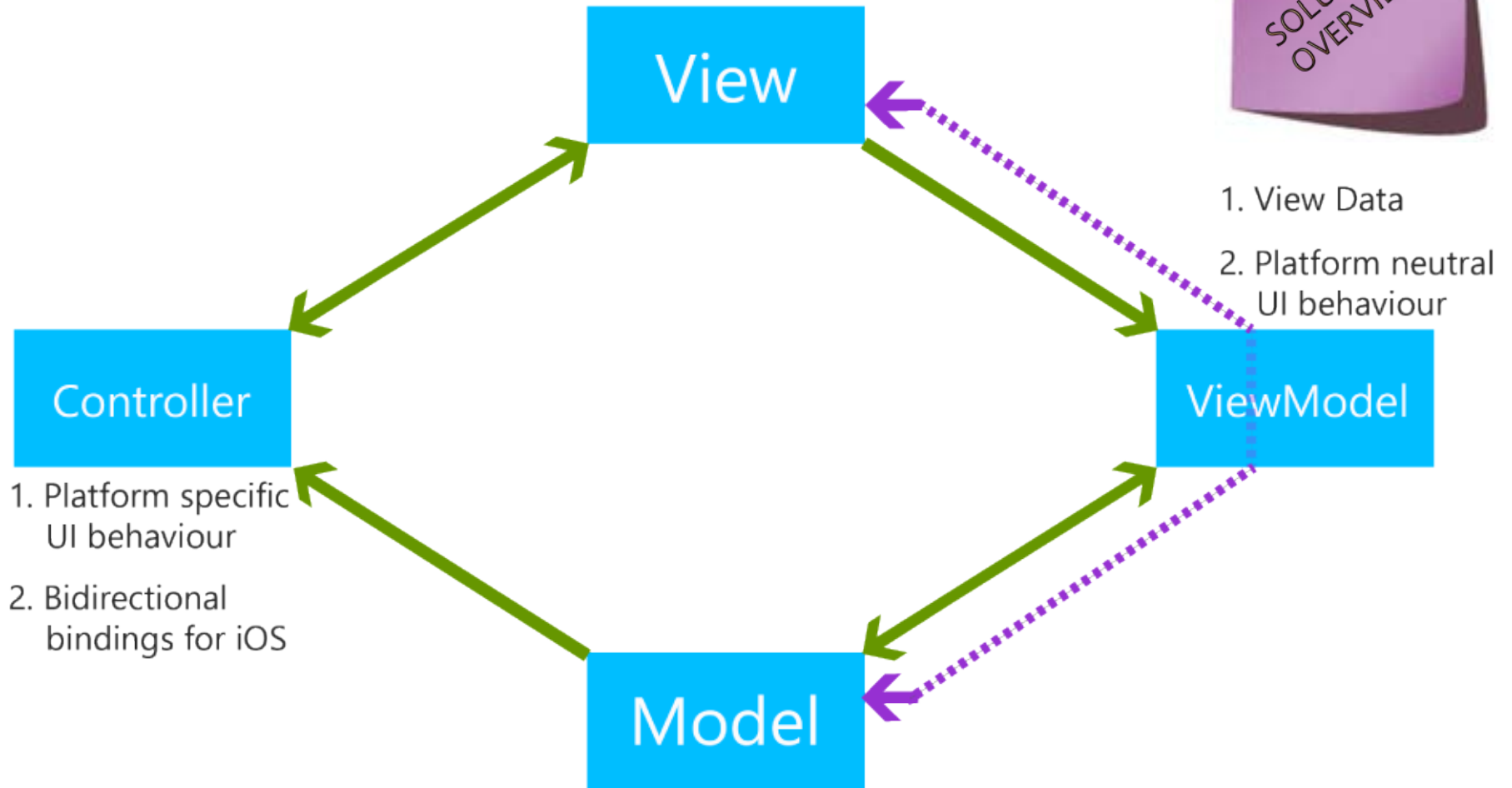
View hooks into
View Model

View Models
are shared

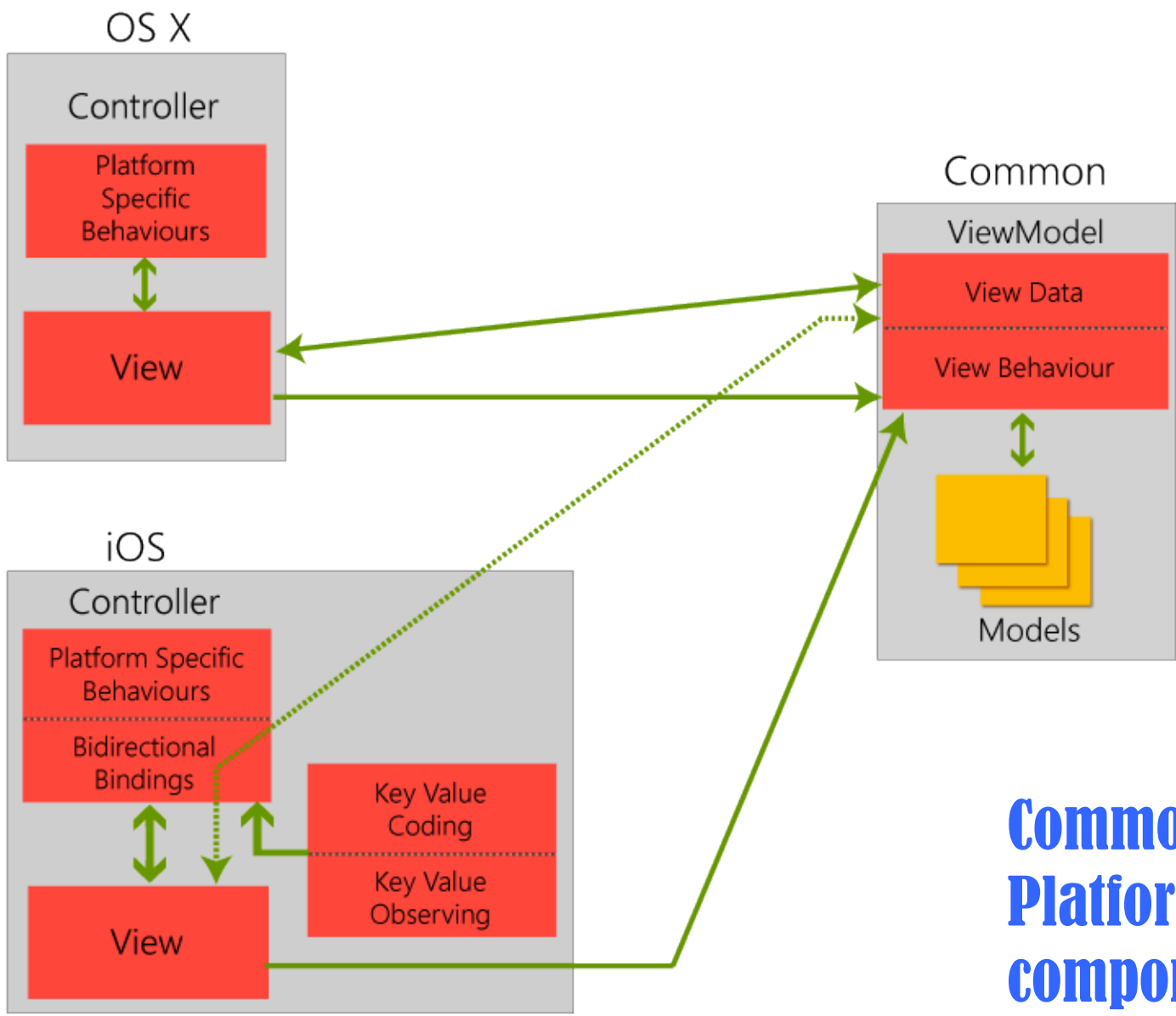
View Models
implement
common part
of UI

Bidirectional
bindings are
platform-specific





- 1 View Models contain view data and common UI behavior
- 2 Models are exposed to Views through View Models



Common & Platform-specific components of the solution



Technology		Literature	
Name	Price (LKR)	Name	Price (LKR)
Advanced Objective C	1586.61	Romeo Juliet	1105.425
Cocoa Programming	1417.545	Odyssey	3251.25
New Book	442.17		

Old		Old	
Name	Price (LKR)	Name	Price (LKR)
Pascal Programming	130.05	Pascal Programming	130.05
Second World War	130.05	Second World War	130.05

Name	Price (LKR)
Pascal Programming	130.05
Second World War	130.05

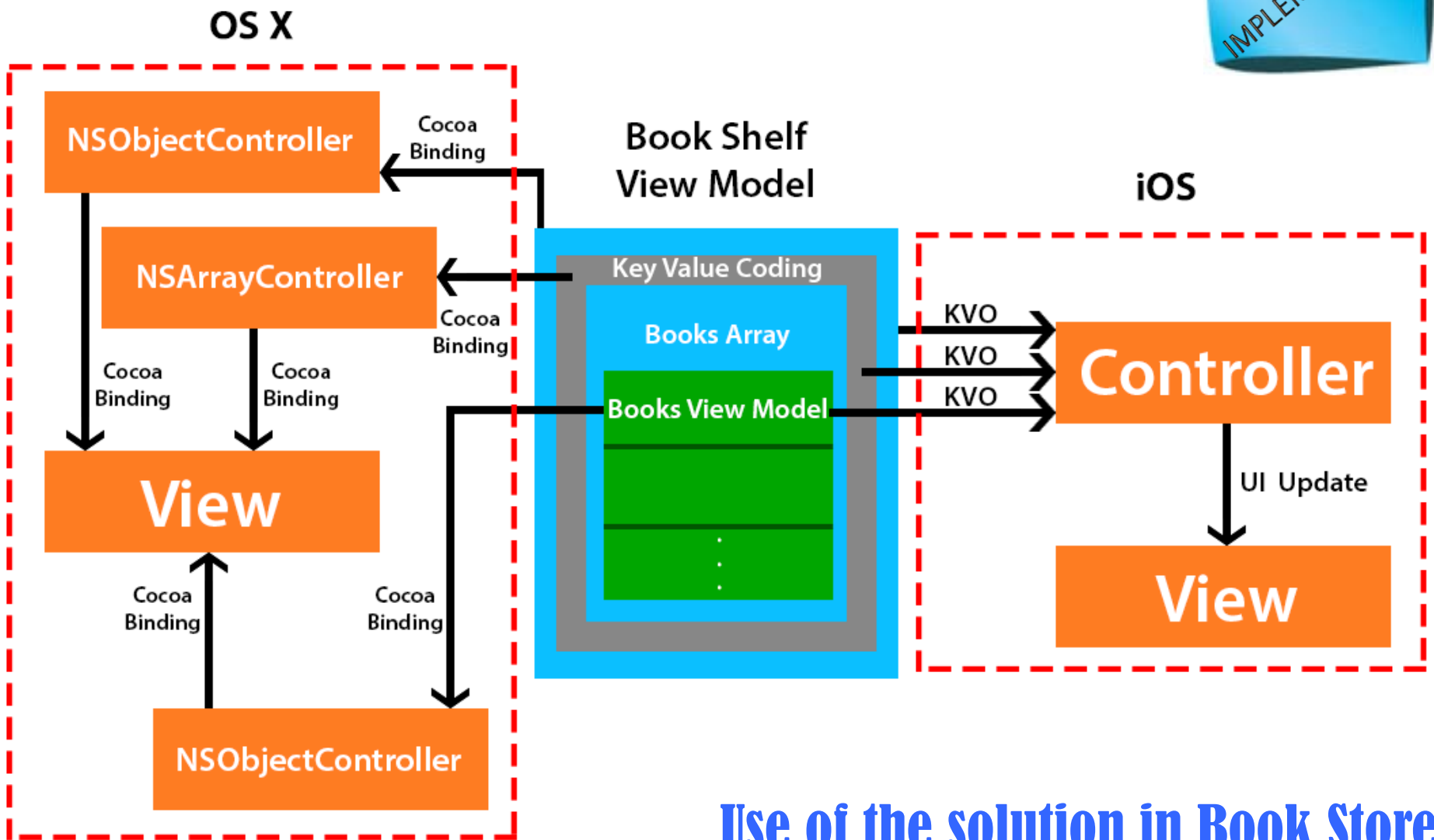
Populate with Source 1

Update

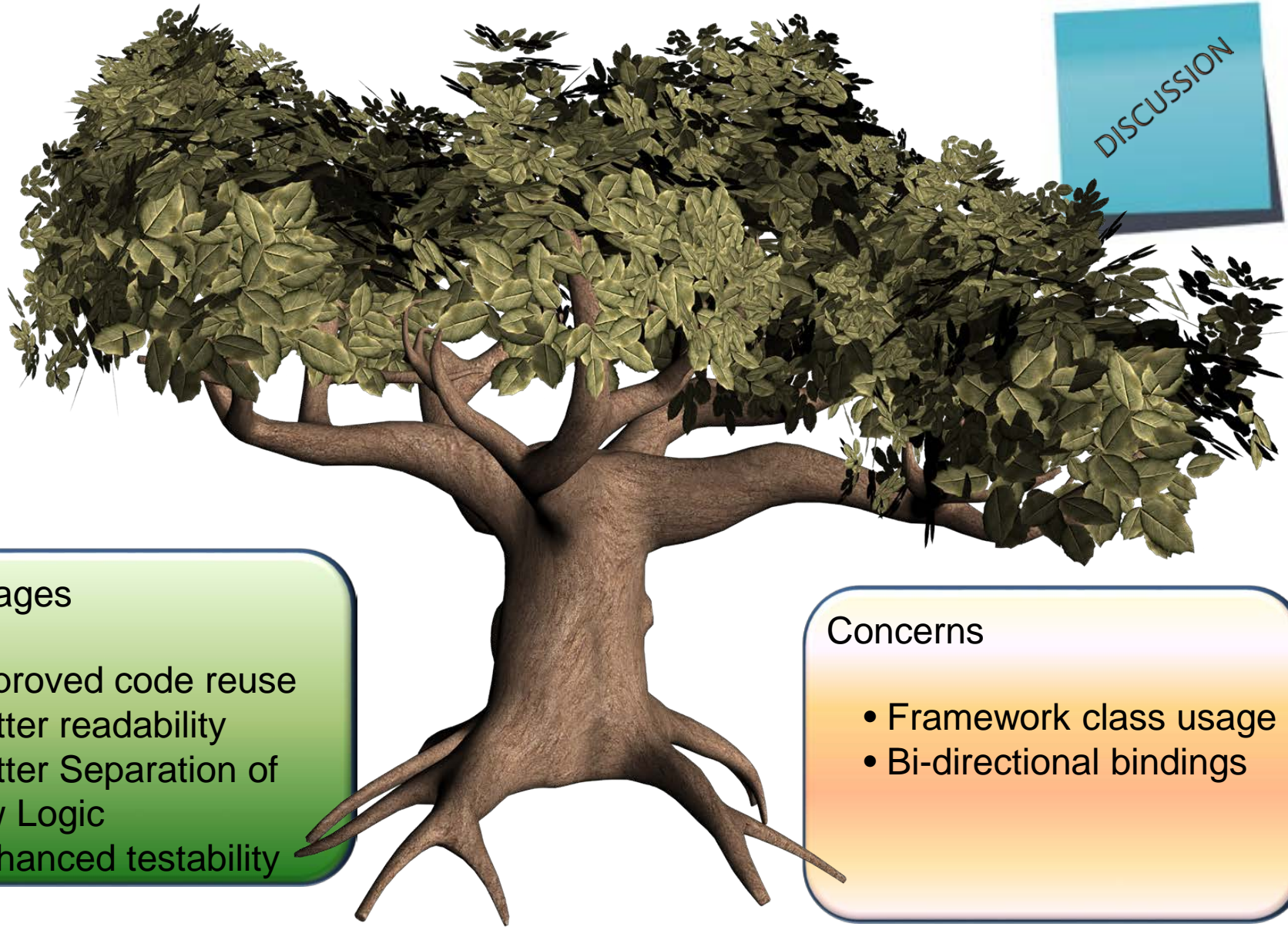
Add new shelf

Populate with Source 2

**Proof of
Concept:
Book Store
Application**



Use of the solution in Book Store application



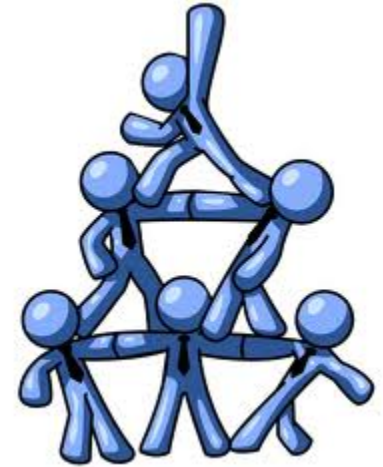
Advantages

- Improved code reuse
- Better readability
- Better Separation of View Logic
- Enhanced testability

Concerns

- Framework class usage
- Bi-directional bindings

CONCLUSION



View Models combined with traditional MVC can improve code reuse significantly while providing other advantages for applications targeting both OS X and iOS.



QUESTIONS