Week 1 - Middle Ware Planning

Carly Kubacek, Ketetha Olengue

1) Explanation and justification of choice of platform used (PHP, Python, etc)

[Product Name]'s application/business logic layer is programmed using the scripting language PHP: Hypertext Processor. PHP is open-source, cheap to use, and easy to debug. Using PHP will lead to a reduced development time for [Product Name] and allow more time for testing.

The main reason for choosing PHP over other scripting languages like Python, Perl, Ruby on Rails, ASP.NET, etc is because PHP works seamlessly with MySQL, the open-source database which supports [Product Name]'s data. The main purpose of a server-side scripting language is to be able to communicate with the database and return results over to the client side. PHP's connection with MySQL makes this an effortless task, and with the availability of third party Database Abstraction Layers, this task becomes even less tedious.

With PHP's shallow learning curve, many other developers in this joint Database Concepts/GUI project will be choosing PHP as well. This local support combined with the global support of developers on the web using PHP makes PHP a well documented scripting language, as opposed to Python. The availability of resources concerning PHP was a big factor in our decision. In addition, PHP offers us an easy transition because of it's foundation based on object oriented programming principles. Many of our developers have worked with object oriented programming in the past using languages like Java, C++, Javascript, etc.

PHP is not only easy to learn and well documented, but flexible as well. It can be used and is well supported on all major operating systems including but not limited to UNIX variants, Linux, Mac OSX, Windows. This allows us more freedom than using a language like Ruby on Rails, which has a better testing and developing environment on Mac OSX than in Windows.

Last but not least, PHP has a wide range of frameworks to choose from. We wanted a language with available frameworks in order to create less work on our end, and still produce a cohesive end-product. These frameworks reinforce

database support, community support, documentation support, and model view controller architecture.

Detailed Explanation of What Functionality Will be Implemented:
(rough, no diagram yet)

User	**Application**	**System (PHP)**
---> create account	fill out form	check if user exists
		create user
		fill in information, save pass
		add to database
---> add photo	get path name	gets current user
		creates photo/add to DB of pht with owner name
		<--- returns success

Classes

User: viewPhoto(path), viewProfile(user), createSearch(query)

Visitor - createAccount()

Administrator - suspendUser(user), deleteComment(commentID, path), deletePhoto(path)

Member - (with an account) signIn(user, pass), signOut(), uploadPhoto(path), tagPhoto(location, user), editPhoto(path) setPrivacy()

Album: create(name), delete(name), insert(photo), display(type)

Photo: addTo(album), hide(), show(), setPrivacy(level), getTags(), deleteTags()

Tags: addTag(path, tag, user), deleteAllTags(path), deleteTagsOfByUser(user)

Comments: addComment(path, user), deleteAllComments(path), deleteAllCommentsOfByUser(user)

Search: searchTag(user), searchAlbum(name), searchPhoto(hashtag)