ADD Process Iteration 1: Establishing an Overall System Structure

This section is a result of following the ADD steps in the first iteration of the design process. This iteration overall will focus on developing a high-level concept design for a system structure after breaking down the greenfield system.

Architectural Concerns

As this is a greenfield system, a select few architectural concerns are identified.

CRN-1: Leverage existing code and team's knowledge to expand and improve existing structure.
CRN-2: Modify existing structure to implement architectural structures and design patterns
CRN-3: Allocate work to all members of the development team.

2. Select Drivers

The following is the first iteration of the ADD process for a greenfield system. The main goal of this iteration is to satisfy CRN-1 which is concerned architecturally with leveraging existing code and knowledge to expand and improve upon an existing structure. Furthermore, because this is a greenfield system, all aspects of the system are drivers as seen in the following:

- QA-2, QA-6 : Usability
- QA-4, QA-5: Performance
- QA-1, QA-7: Security
- QA-3: Scalability
- CON-1: A minimum of 4 simultaneous users must be supported
- CON-2: The system must be accessed through a web browser on Windows
- CON-3: Listings are stored for a maximum of 30 days before being archived
- CON-4: The server for the database must be generally reliable (Up when needed for demonstrations)

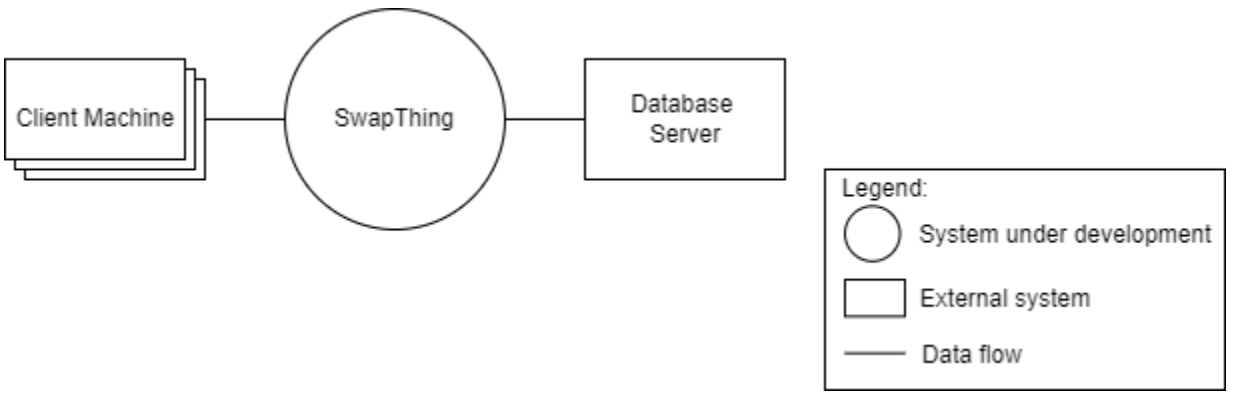

Figure 1 : Iteration 1 Context Diagram

3. Select Elements to Refine

Although this is a pre-existing system, the application has been treated like a greenfield development project to ensure all steps of the ADD process are thoroughly implemented from the start of the development process.

4. Choose Design Concepts to Satisfy Drivers

The following tables describe all the chosen and rejected design concepts that are relevant to the system and its drivers.

Design Decisions and Location	**Rationale/Justification**
Logically structure the client part of the system using the web applications architecture pattern	- This reference architecture supports little business logic to be hosted on the client side. (supports ALL UC, CON-2, QA-2, QA-6) - Applications are heavily serverside and do not rely on client to validate
Integrate logical components of Service Applications to implement pages and facilitate different content display	The web application architecture pattern overrides the non-interactive component of this application, however, service layers generate and provide content which is formatted to be sent to a user. Minimizing the input required from the user supports QA-4, QA-5. Templates are used then populated with data from the data store.
Physically structure the application using the Three-Tier Deployment pattern	The system will be accessed from a web browser meanwhile a separate server and database will be established (CON-2 and CON-4). This requires the use of a three-tier deployment pattern as any n<3 or n>3 tier deployment pattern either does not satisfy the system's requirements or is unnecessary for the system.

REJECTED Design Decisions and Location	**Rationale/Justification**
Logically structure the client part of the system like a rich internet application architecture pattern	For security and data access purposes the server side manages the creation of pages.
Logically structure the client part of the	While the client side requires service and

system like a service application architecture	data from the server side this architecture pattern does not allow for deep user interaction required for this system.

5. Instantiate Architectural Elements, Allocate Responsibilities, and Define Interfaces

The following table describes the design decisions made for instantiating certain elements of the system architecture.

Design Decisions and Location	Rationale
Remove local data sources in the web application	Server/Database connection is hosted by a third party with reliable service, making it unnecessary for app data to be stored locally
The interface layer of the service architecture will be responsible for page creation specifically viewing listings, user profiles, and offers.	Each page is custom made to be formatted and contain information relevant and sometimes private to each user. It is necessary to have pages created server side to maintain security and prevent user access to restricted data.

The next step will show a representation of these instantiation design decisions. As this is the first iteration, the goal for now is to only create high-level definitions for components and their functionality and interfaces. A more granular explanation of each component will be provided in the next iteration.

6. Sketch Views and Record Design Decisions

The following sketch shows a component view of both the client and server side applications and their respective reference architectures. This can now be visually represented after making the necessary design decisions.

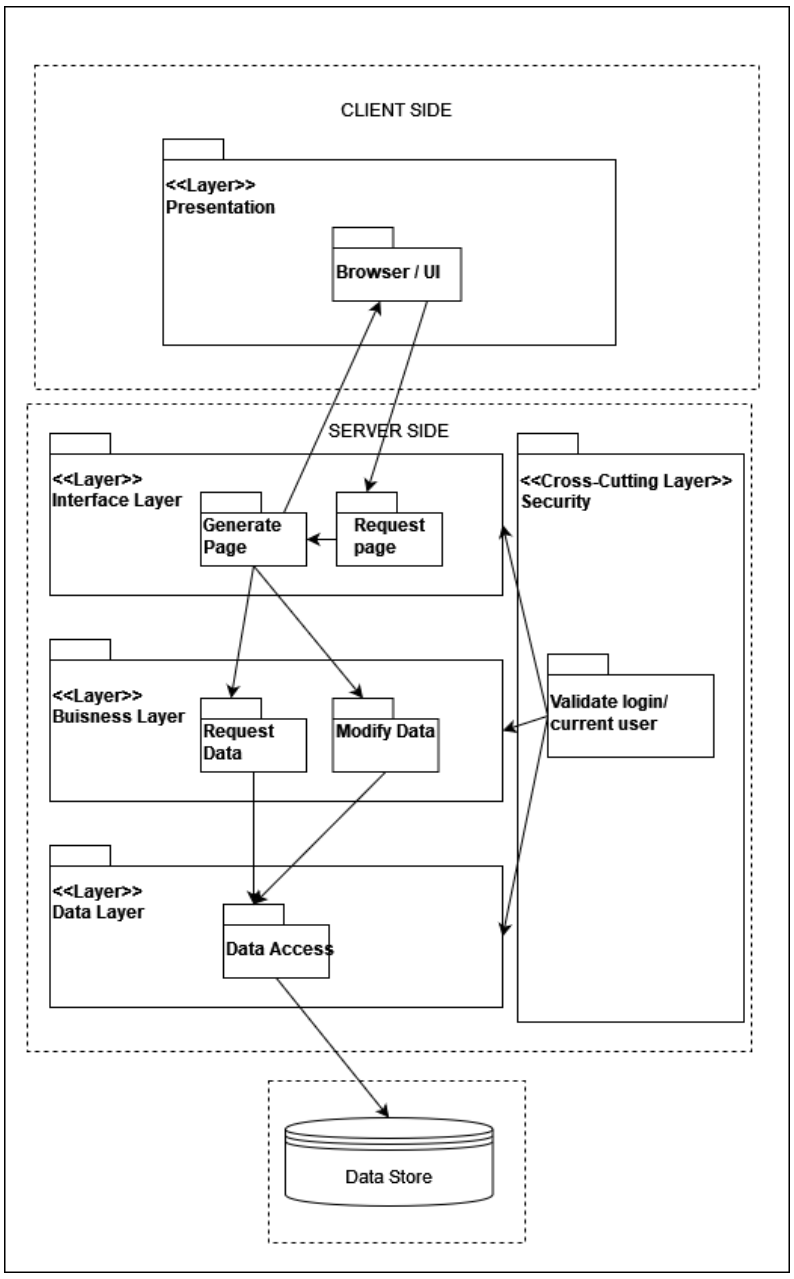

Figure 2 : Iteration 1 component-view sketch

As a note to keep in mind, the component-view sketch is a very high-level visual representation of the system describing just major components and their responsibilities, the following table further details each component and their responsibilities.

Element	Responsibility
Presentation client side	This layer contains components that control user interaction and use case control flow.
Browser/UI	These components render the user interface and receive user inputs. These components are responsible for control flow of all the system use cases (including navigation between screens)
Interface Layer server side	These components expose services that are consumed by the clients.
Generate Page	This component is responsible for creating the necessary pages for the presentation layer.
Request Page	This component is responsible for sending a message with details on which page to create to be sent to the presentation layer.
Business Layer server side	This layer contains components that perform business logic operations that require processing on the server side. These components implement business operations.
Request Data	This component is responsible for sending a message to receive data to be sent to the interface layer.
Modify Data	This component is responsible for sending a message to request permission to modify data.
Data Layer server side	This layer is responsible for requests of permission to view/modify the remote database to retrieve the necessary data to go back upstream and ultimately be viewed by the client.
Data Access	This component is responsible for persistence of business entities (objects) into the relational database. It performs object-oriented to relational mapping and

	shields the rest of the application from persistence details.
Cross Cutting Layer SS	These components have functionality that goes across different layers, such as security, logging in, and IO.
Validate login/Current User	This component is responsible for ensuring the username and password for each login attempt is valid, also for keeping track of the current user.
Data Store	This component is responsible for supplying the mass storage of most of the system's data.

The following diagram illustrates how the previously described components and their associated components will be deployed and interact with each other.

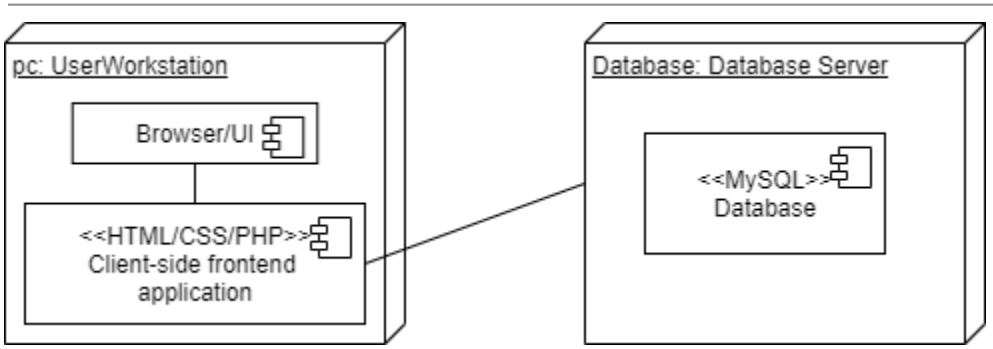

Figure 3 : Iteration 1 Initial Deployment Diagram

The following table details the responsibility of each component:

Element	Responsibility
User Workstation	Hosts the client side logic of the system
Database Server	Hosts the remote database

Furthermore, the following table describes the relationship between these components:

Relationship	Description
Web application and database server	Communication between the two components will be done using PHP and MySQL

7. Perform Analysis of Current Design and Review Iteration Goal and Achievement of Design Purpose

The following table details and describes the design process using the Kanban board technique.

Not Addressed	Partially Addressed	Completely Addressed	Design Decisions Made During Iteration
QA-3	QA-1	CON-2	Logically structure the client part of the system using the web applications architecture pattern
QA-3	QA-6		Integrate logical components of Service Applications to implement pages and facilitate different content display
QA-3			Physically structure the application using the Three-Tier Deployment pattern
CON-4 QA-4		QA-1	Remove local data sources in the web application
	QA-2	QA-6	The interface layer of the service architecture will be responsible for page creation specifically viewing listings, user profiles, and offers.

Quality Attributes, Constraints not addressed/applicable in this iteration:
QA-5, QA-7, CON-1, CON-3