Software Design & Architecture: Project Deliverable 3, Iteration 1

Taimour Arshad - 100748446

Raza Naqvi - 100754516

Massimo Albanese - 100616057

Waleed El Alawi - 100764573

Date: November 12, 2021

Iteration 1

ADD Step 1: Reviewing Inputs

<table>
<tr><th>Category</th><th>Details</th></tr>
<tr><td>Design Purpose</td><td>This is a greenfield system with no prior architectural concerns or liabilities. The purpose is to produce a sufficiently detailed design to support the construction of the system.</td></tr>
<tr><td>Primary functional requirements</td><td>- UC-1: Integral to the functionality of the product
- UC-3: Integral to the functionality of the product
- UC-4: Integral to the functionality of the product
- UC-5: Integral to the functionality of the product
- UC-6: Integral to the functionality of the product</td></tr>
<tr><td>Quality Attribute Scenarios</td><td><table>
<tr><th>Scenario ID</th><th>Importance to the Customer</th><th>Difficulty of Implementation</th></tr>
<tr><td>QA-1</td><td>High</td><td>High</td></tr>
<tr><td>QA-2</td><td>High</td><td>Medium</td></tr>
<tr><td>QA-3</td><td>High</td><td>Medium</td></tr>
<tr><td>QA-4</td><td>Low</td><td>Low</td></tr>
</table></td></tr>
<tr><td>Constraints</td><td>All constraints discussed are included as drivers.</td></tr>
<tr><td>Concerns</td><td>All concerns discussed are included as drivers.</td></tr>
</table>

ADD Step 2: Establish Iteration Goal by Selecting Drivers

To achieve the Iteration Goal CRN-1 of *Establishing an Overall Initial System Structure*, the selected drivers needed are as follows:

- QA-1: Security
- QA-4: Performance
- CON-1: The system must support at least 1000 users
- CON-2: The calendar software must be coded in Python, JavaScript, HTML, CSS

ADD Step 3: Refine System Elements

As this is a greenfield system, the element to refine would be the entire social calendar system. Refinement will be performed through decomposition.

ADD Step 4: Select One or More Design Concepts that Satisfy the Selected Drivers

<table>
<tr><th>Design Decisions and Location</th><th>Rationale</th></tr>
<tr><td>Logically structure the client part of the system using the <b>Rich Client Application</b> reference architecture</td><td>The <b>Rich Client Application (RCA)</b> reference architecture supports the development of applications that are installed on the users’ PC. These applications support rich user interface capabilities that are needed for displaying the users’ calendars (UC-3), their friends’ calendars (UC-5), and multiple calendar views (UC-7).

<b>Discarded alternatives:</b>
<table>
<tr><th>Alternative</th><th>Reason for Discarding</th></tr>
<tr><td>Rich Internet applications (RIA)</td><td>This reference architecture is geared towards the development of applications with a rich user interface that runs inside a web browser. This option was discarded because we want to have an application installed on the users’ computer for usability and convenience.</td></tr>
<tr><td>Web applications</td><td>This reference architecture is oriented toward the development of applications that are accessed from a web browser. This reference architecture was discarded because it is too difficult to provide a user rich interface.</td></tr>
</table>
</td></tr>
<tr><td>Logically structure the server part of the system using the <b>Service Application</b> reference architecture</td><td>This backend server would provide an API for the client application. No other alternatives were considered or discarded, as the architects were familiar with the reference architecture and considered it fully adequate to meet the requirements.</td></tr>
<tr><td>Physically structure the application using the <b>three-tier deployment</b></td><td>Since the system would have a client application, a backend server, and would need to store user and performance data (UC-1, UC-3, UC-4, UC-6, UC-8), a three-tier deployment is appropriate as a database server would be needed to store this information.</td></tr>
</table>

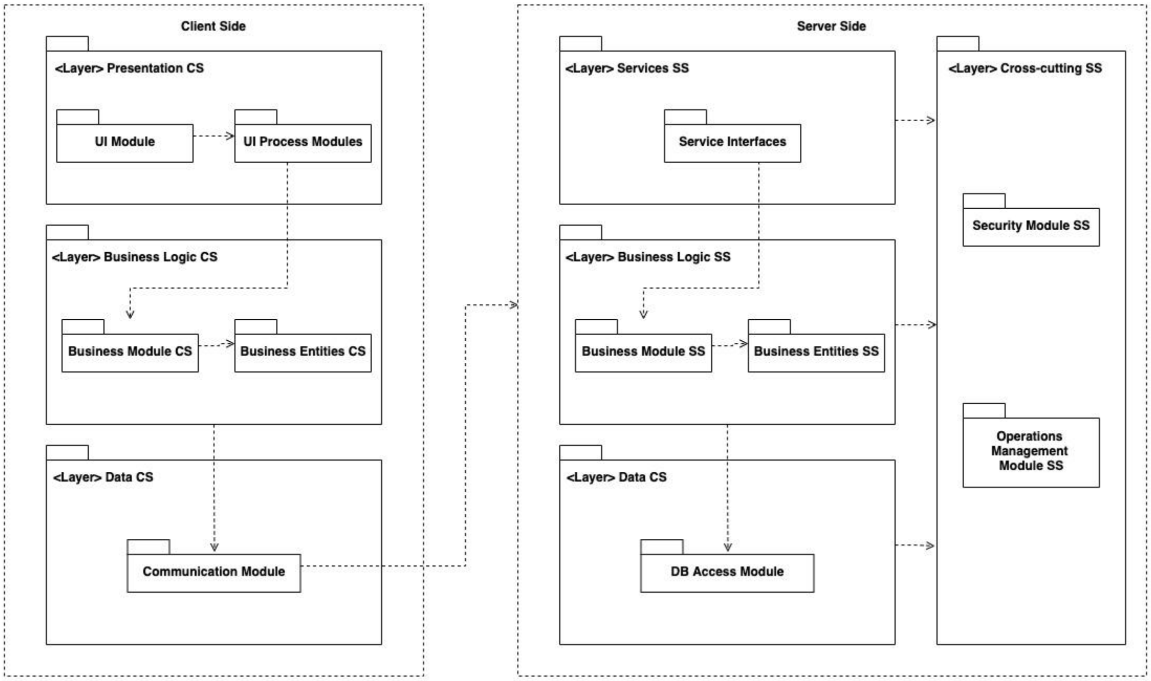
pattern	
Build the client application using the Electron framework	The Electron framework allows for the creation of desktop GUI applications using web technologies, providing all of the benefits of a Rich Internet Application and a Web Application, while being locally installed on a user’s machine.
Build the user interface of the client application using web technologies (HTML/CSS/JS)	Building the user interface with HTML, CSS, and JavaScript is what all the developers are already familiar with (CON-2).
Build the server part of the system using the FastAPI framework (Python)	Building the backend server system using the FastAPI framework would allow for many concurrent users as it is an asynchronous system (CON-1), and would leverage the existing experience and knowledge of the technology by the developers (CON-2).

ADD Step 5: Instantiate Architectural Elements, Allocate Responsibilities, and Define Interfaces

It is too early to decide how we precisely want to define the functionalities and interfaces. We will go into further detail in the next iteration.

ADD Step 6: Sketch Views and Record Design Decisions

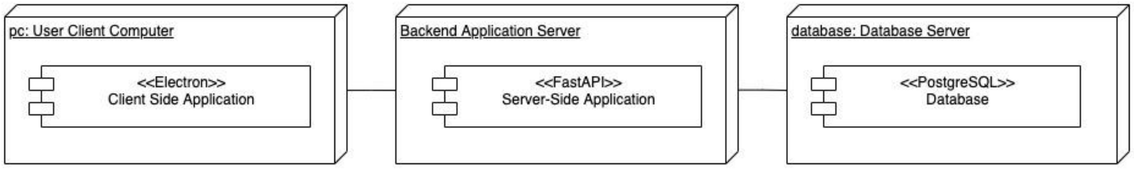
Architecture Model View



Element	Responsibility
Presentation CS	This layer is responsible for user interaction and use case control flow.
Business Logic CS	This layer contains modules that perform business logic operations on the client side.
Data CS	This layer is responsible for communicating with the server.
Services SS	This layer contains modules that show services that are consumed by the clients
Business Logic SS	This layer contains modules that perform business logic operations on the server side
Cross-cutting SS	Contains modules that cross different layers, like the security module.
UI Modules	These modules render the user interface and receive user inputs.

UI Process Modules	These modules are responsible for control flow of all the system use cases (including navigation between screens).
Communication Module	These modules utilize the services provided by the application on the server side.
Business Modules CS	Modules that implement business operations or they expose business functionality from the server side.
Business Entities CS	These entities make up the domain model.
Service Interfaces	These modules expose services that are consumed by the clients
Business Modules SS	These modules implement business operations.
Business Entities SS	These entities make up the domain model.
DB Access Module	This module is responsible for persistence of business entities into the relational database.
Security Module SS	This module is responsible for controlling access to and from the application.
Operations Management Module SS	This module is responsible for accessing, diagnosing, and documenting interactions between layers.

Initial Deployment Diagram



ADD Step 7: Analysis of Current Design, Iteration Goal Review, and Achievement of Design Purpose

Not Addressed	Partially Addressed	Completely Addressed	Design Decisions Made During Iteration
	UC-1		Selected reference architecture establishes the modules that will support this functionality.
	UC-3		Selected reference architecture establishes the modules that will support this functionality.
	UC-4		Selected reference architecture establishes the modules that will support this functionality.
	UC-5		Selected reference architecture establishes the modules that will support this functionality.
	UC-6		Selected reference architecture establishes the modules that will support this functionality.
QA-2			No relevant decisions made.
QA-3			No relevant decisions made.
	QA-4		Selected reference architecture establishes the modules that will support this functionality.
	CON-1		Selected reference architecture establishes the modules that will support this functionality. Decisions regarding handling of concurrent access have not been made yet.
		CON-2	The decided overall system structure leverages the development team’s prior experience and skills.
CON-3			No relevant decisions made.
		CRN-1	Reference architecture and deployment pattern have been selected to address this concern.
CRN-2			No relevant decisions made.