Packagebird Alpha Prototype

CPT_S 421 – Dr. Bolong Zeng

Elisha Aguilera, Setenay Guner, Denish Oleke

Mentored by Brandon Busby

Date Submitted: 12-10-2021

Alpha Prototype Description – 1

Features implemented in the Packagebird Alpha Prototype:

- Configuring development environment
    - Downloading packages from server
    - Formatting Packages in directory
- Running rudimentary test on packages
- Running rudimentary builds on packages
- Getting a Command Line 'Help' list
- Creating packages from projects

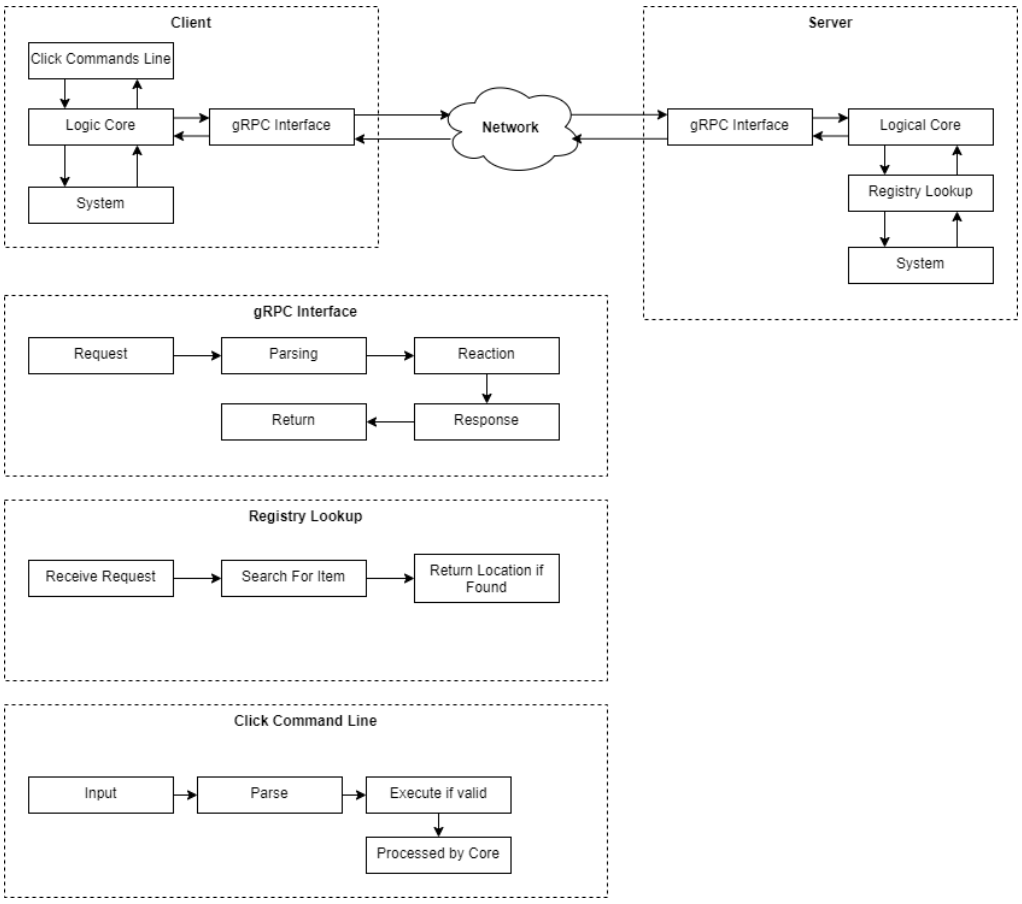

Figure 1-1: Diagram illustrating interactions between different components, the primary path being an initiation from a client command line application to a server.

As illustrated in Figure 1-1, the Alpha Prototype of Packagebird performs operations primarily initiated by a commend line application client on a developer's machine, then interacting with the configured remote server. On the client, the developer initiates the example process of configuring the local environment with a command, parsed through the command interface

component, and if validated progresses to the logical core before finally being converted to a network request to the server. On the receiving end, the server takes the network request, and depending on the contents processes the request. The most common cases will be looking up requested packages or projects and if found transmitting them back to the client, the other common process being the creation of new packages from the developer's machine. A majority of these operations will have a response from the server to the client, which will depending on the operation acknowledge and proceed with another operation.

Operations are small but often can happen in various orders depending on context: if a user attempts to create a package containing nothing, the command line will deliver a warning to the user if they wish to proceed, and if approved will perform the task. If a user desires, they can first configure a directory from a project, then initiate the package creation, as intended.

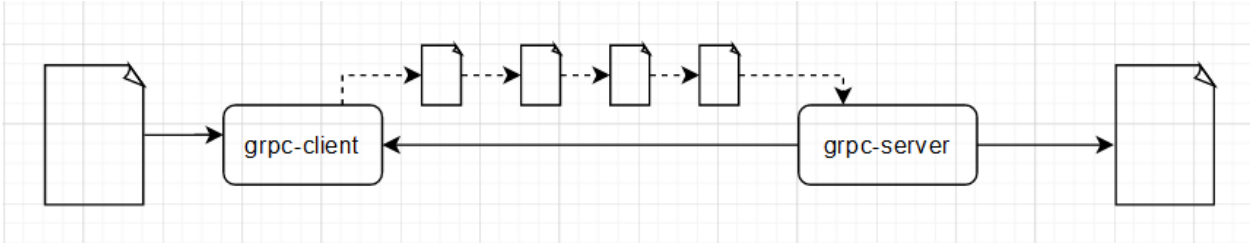

Figure 1-2: An exchange of files between the client and the server.

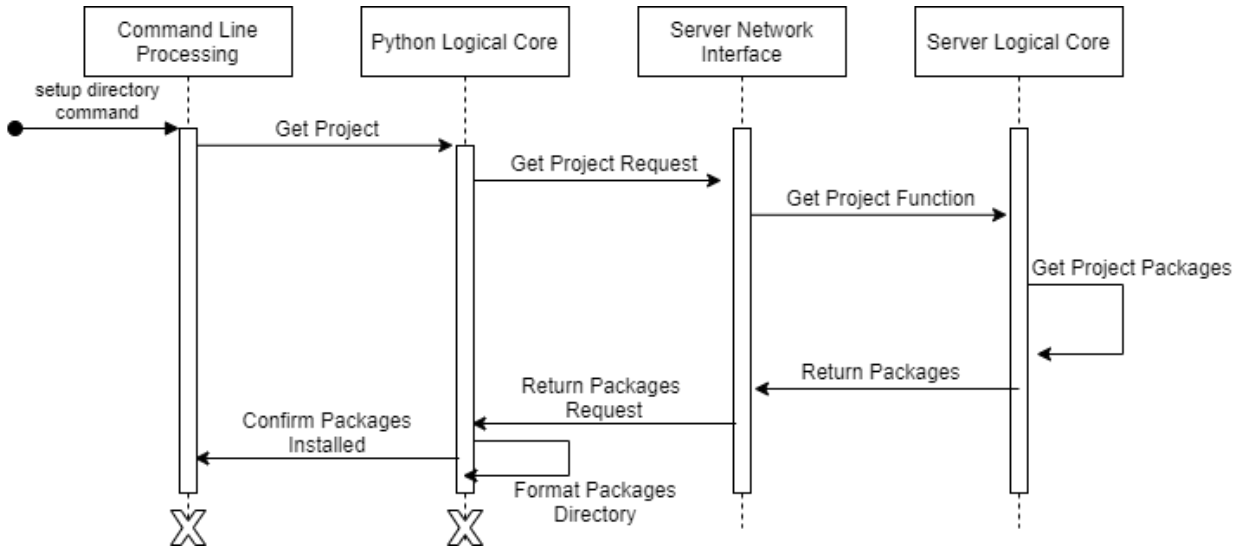

Figure 1-3: Sequence diagram of a project setup process.

Modifications Stemming from Alpha Prototype – 2

The current iteration of Packagebird makes heavy reliance of relative paths for installing packages. Research and feedback have indicated that relative paths may be considered a security vulnerability if a rogue user where to use specially formatted package.

We are currently running the SQLite database within our server as a lightweight option. It is easy to implement as well as providing fast read-write operations. One of the main reasons this was our first choice because it features zero-configuration.

One of the considerations that was brought to our attention by our stakeholder was relational versus document databases for this project. While Relational offers accuracy and consistency they work best with structured data. They also do not scale well. On the other hand, working with NoSQL gives us an advantage of storing large amounts of data with little structure. The two options are equally useful in their own way but for contrasting use-cases.

Future of the Project – 3

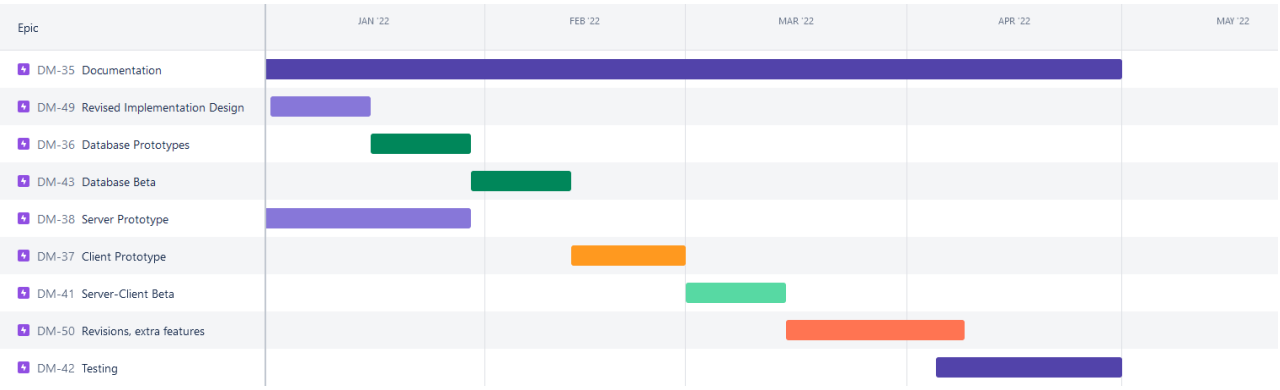

Figure 3-1: Gantt Chart from Jira board

[Link to chart on JIRA]

For the beta version we can split development into several stages, first the database part which we will be implementing right after our break for storing packages and projects. The second part, we will be focusing on client-server communication which is a crucial part of the project. Client will be able to create, remove and edit packages and projects. As we get closer to deployment we will also be working on extra features, such as dependency graphs and web interface. Finally, the project will enter a testing stage where we will be applying various test cases to different units and features.

Judging from the first half of the project and as with all projects, the original timeline might progress over time and child issues might be rearranged, however, our current sprint plans are seen below.

Epic	Child Issue	Assigned	Start	End
Documentation	Beta Prototype Updates	All	-	4/30
	Midterm Progress Report			
	Final Report			
	Software Document			
Revision of Implementation	Security	Eli	01/5	01/15
	Re-evaluate stakeholder needs	Set & Denish		
	Feedback	All		
Database Prototype	Relational vs Document	Set & Eli	1/16	1/29
	Research Solutions	Denish		
	Evaluate Solutions	All		
Database Beta	Implementing DB Solution	Eli	1/30	2/12
	CRUD functions on DB	Set		
	Mock Client	Eli		

Table 3-1: Task Table for Spring 2022 Semester

Epic	Child Issue	Assigned	Start	End
Server Prototype	Re-structuring gRPC	Denish	12/22	2/5
Client Prototype	Create Packages	Set	2/13	2/28
	Create Projects	Denish		
	Credentials	Eli		
Server-Client Beta	Operations Exchange Implementation	All	3/1	3/14
Revision, Features	Revise Beta – Feasibility for extra features	All	3/15	4/8
	Client Authentication	Eli		
	Dependency Graph	Set & Denish		
Testing	Perform system testing	Eli & Set	4/8	4/30
	Verify & Validate	Denish		
	Revise Documents	Denish		

Table 3-1 (Continued): Task Table for Spring 2022 Semester