Name of Student: Pushkar Sane		
Roll Number: 45		**Lab Assignment No: 6**
Title of LAB Assignment: To develop application using Google App Engine by using Eclipse IDE. Find the position of a target value within a sorted integer array. (Binary Search)		
DOP: 14-10-2024		**DOS: 17-10-2024**
CO Mapped: CO6	**PO Mapped: PO1, PO2, PO3, PO7, PO9, PSO1**	**Signature:**

Practical No. 6

Aim: To develop application using Google App Engine by using Eclipse IDE. Find the position of a target value within a sorted integer array. (Binary Search)

Description:

Platform as a Service (PaaS) is a cloud computing service model that provides a platform allowing developers to build, run, and manage applications without worrying about the underlying infrastructure. PaaS supplies hardware and software tools over the internet, typically requiring the user to manage the applications but not the servers or storage. It simplifies the application development process by offering pre-built infrastructure and deployment environments, allowing developers to focus on coding rather than managing the back-end systems.

Key Features of PaaS:

1. **Development Frameworks**: Provides frameworks to help developers create cloud-native applications quickly.
2. **Middleware**: PaaS comes with middleware that integrates with databases and other services.
3. **Infrastructure Management**: It automates much of the infrastructure management, such as load balancing and scaling.
4. **Development Tools**: Offers built-in tools like version control systems, debugging, and testing tools.
5. **Scalability**: PaaS platforms scale applications automatically based on the traffic load.

Example: Google App Engine

Google App Engine (GAE) is a well-known PaaS offering by Google that enables developers to build and host web applications on Google-managed infrastructure. It abstracts much of the infrastructure complexity, allowing developers to focus solely on writing code.

Features of Google App Engine:

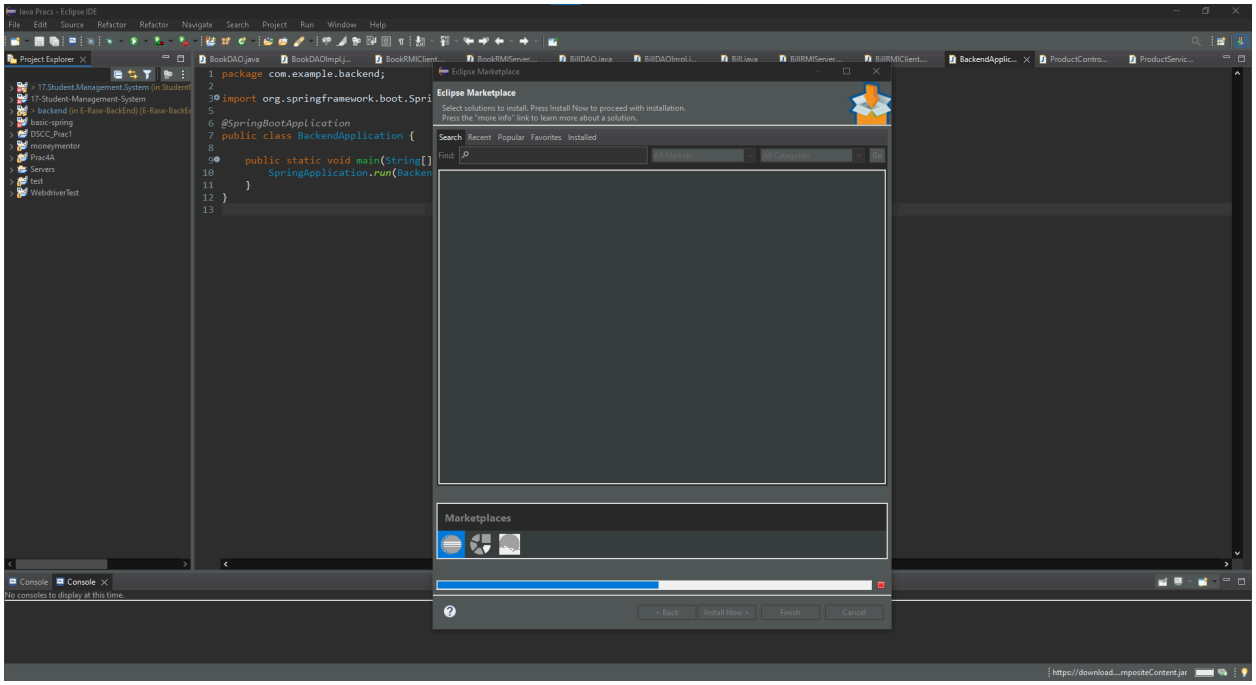
1. **Automatic Scalability**: GAE automatically scales applications up or down based on traffic.
2. **Managed Platform**: It handles server management, such as configuration, load balancing, and scaling, for the user.
3. **Multiple Languages**: Supports various programming languages like Java, Python, Go, Node.js, PHP, and Ruby.
4. **Integrated Google Services**: Offers easy integration with other Google Cloud services such as Datastore, Cloud SQL, and Cloud Storage.
5. **Flexible Deployment**: GAE supports both standard and flexible environments, allowing developers to choose between sandboxed or container-based environments.

Google App Engine is ideal for building scalable web applications and services, eliminating the need to manage infrastructure manually while leveraging Google's extensive cloud resources.

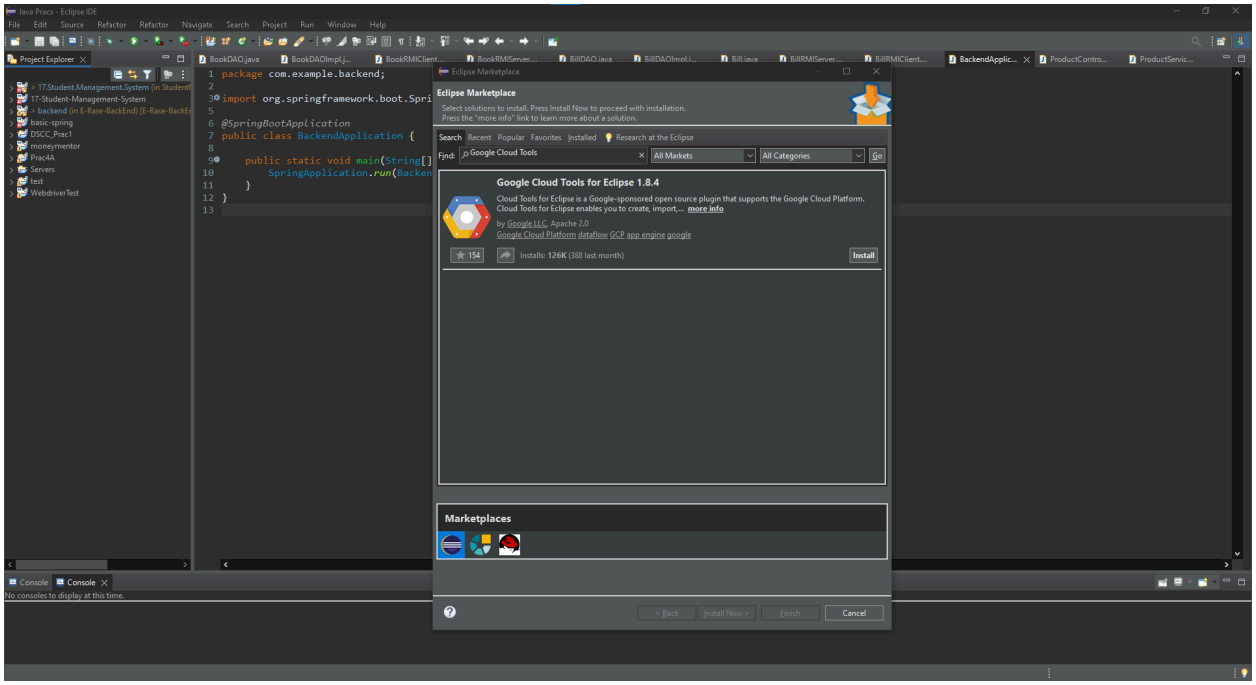
To develop application using Google App Engine by using Eclipse IDE. Find the position of a target value within a sorted integer array. (Binary Search)

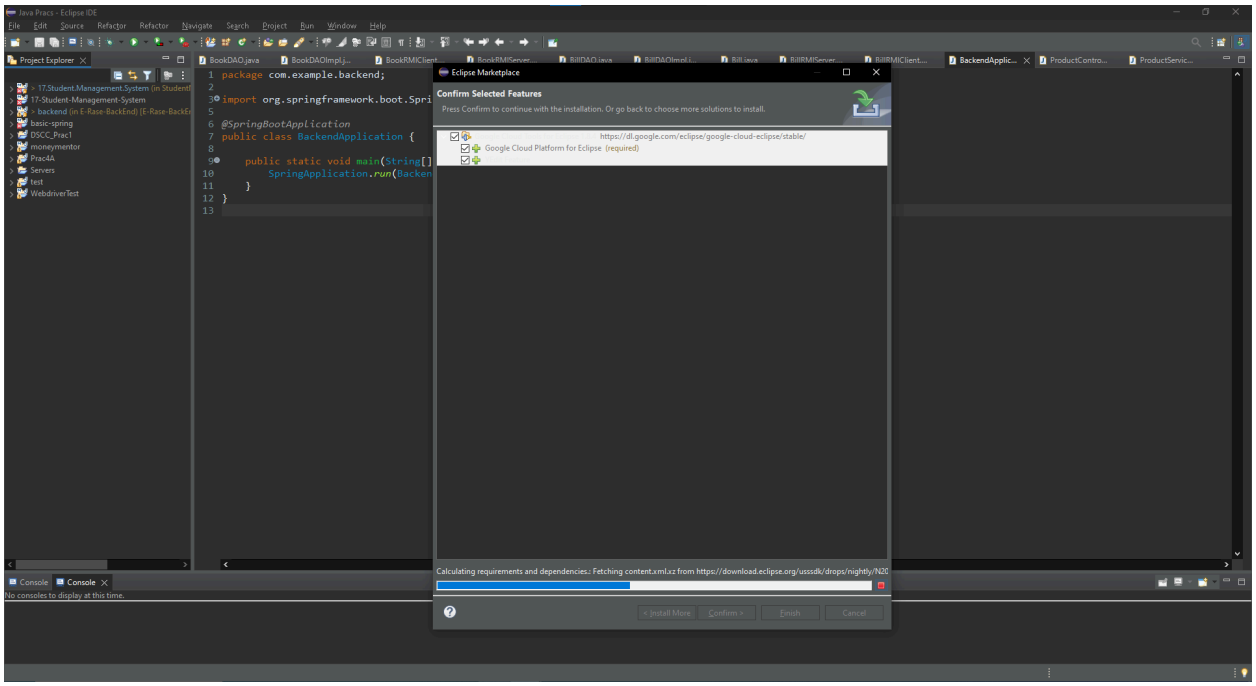
Step 1: Set Up Your Google App Engine Project

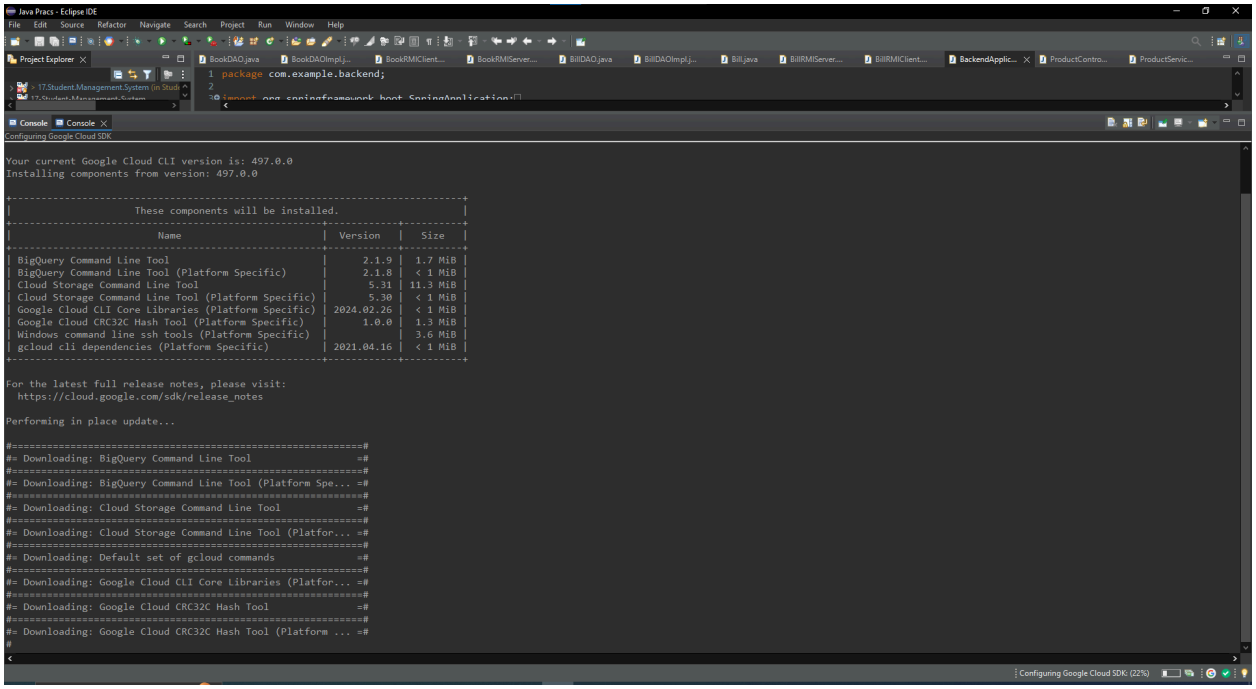
1. **Install Eclipse IDE**: Ensure you have Eclipse IDE installed. You can download it from the Eclipse website.
2. **Install Google Cloud Tools for Eclipse**: This plugin helps to develop and deploy applications to Google App Engine.
    - Go to Help -> Eclipse Marketplace.



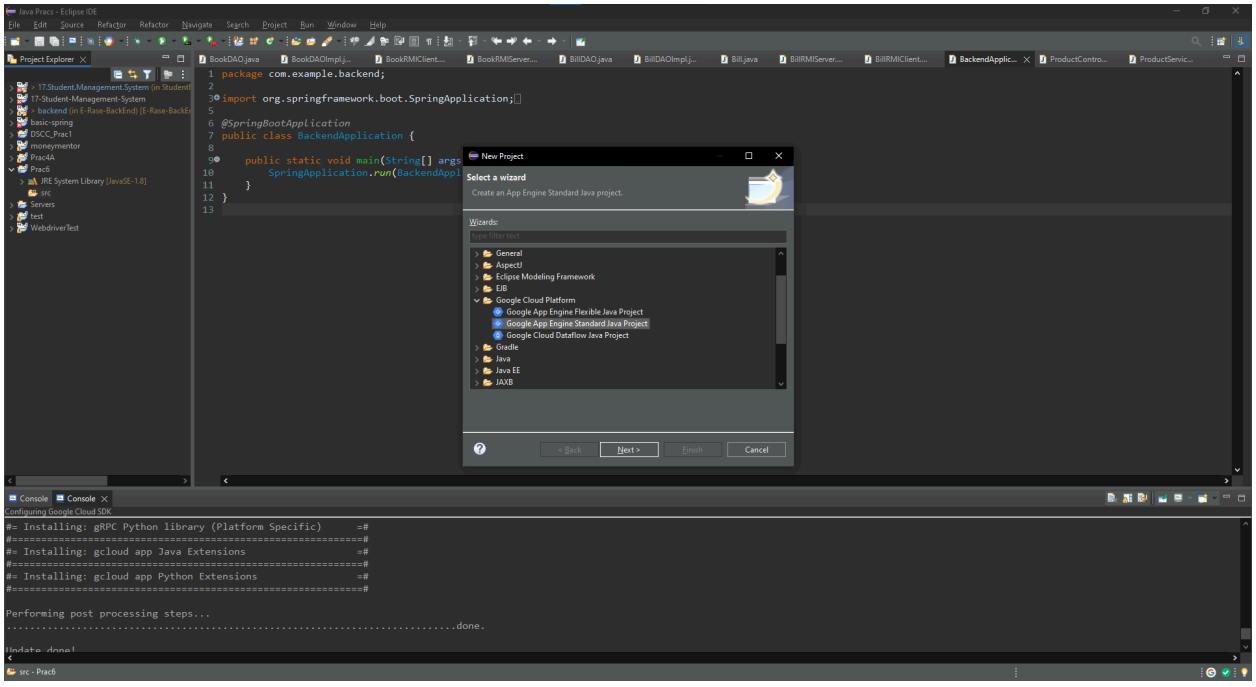
- Search for "Google Cloud Tools" and install it.

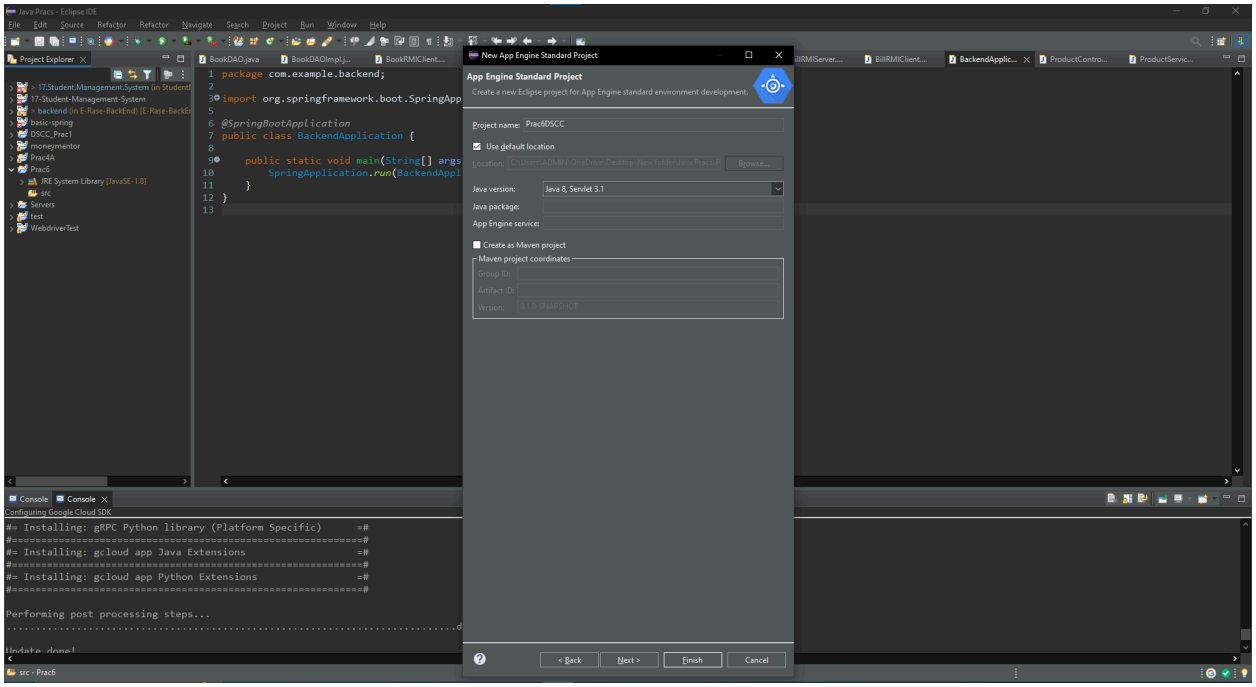






3. **Create a New Project**:
    - Click on File -> New -> Project.
    - Select Google Cloud Platform -> Google App Engine Standard Java Project.

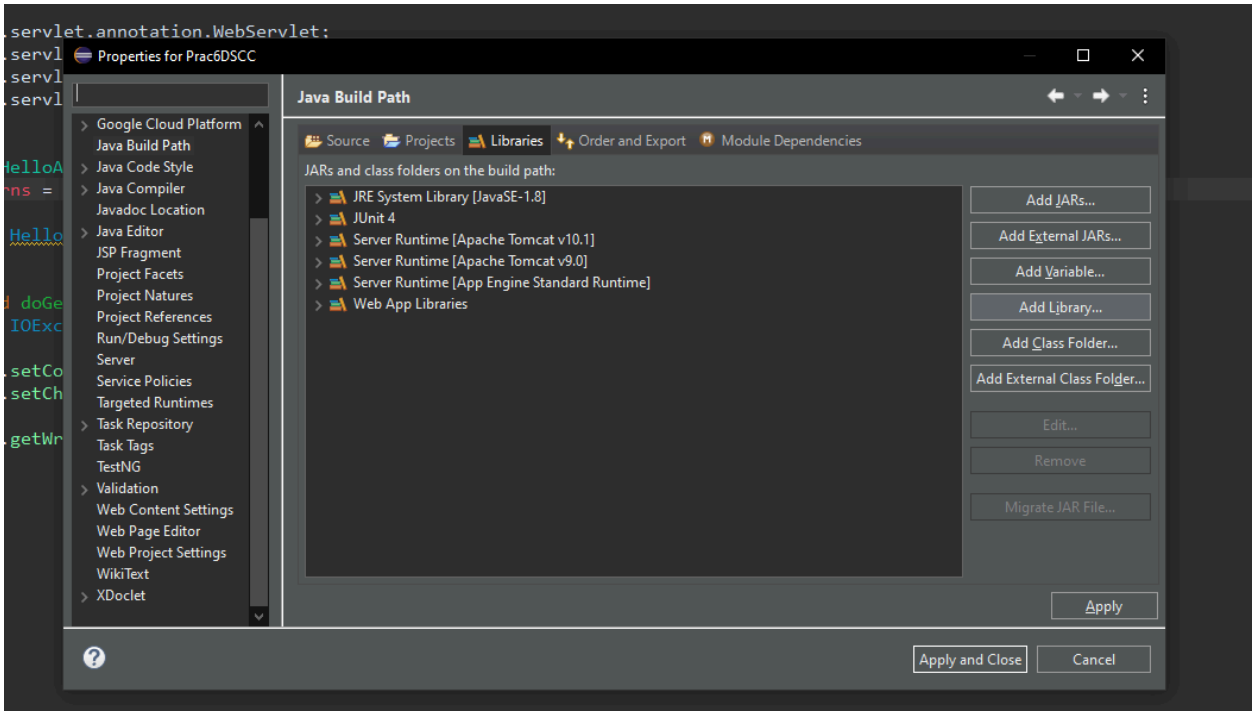




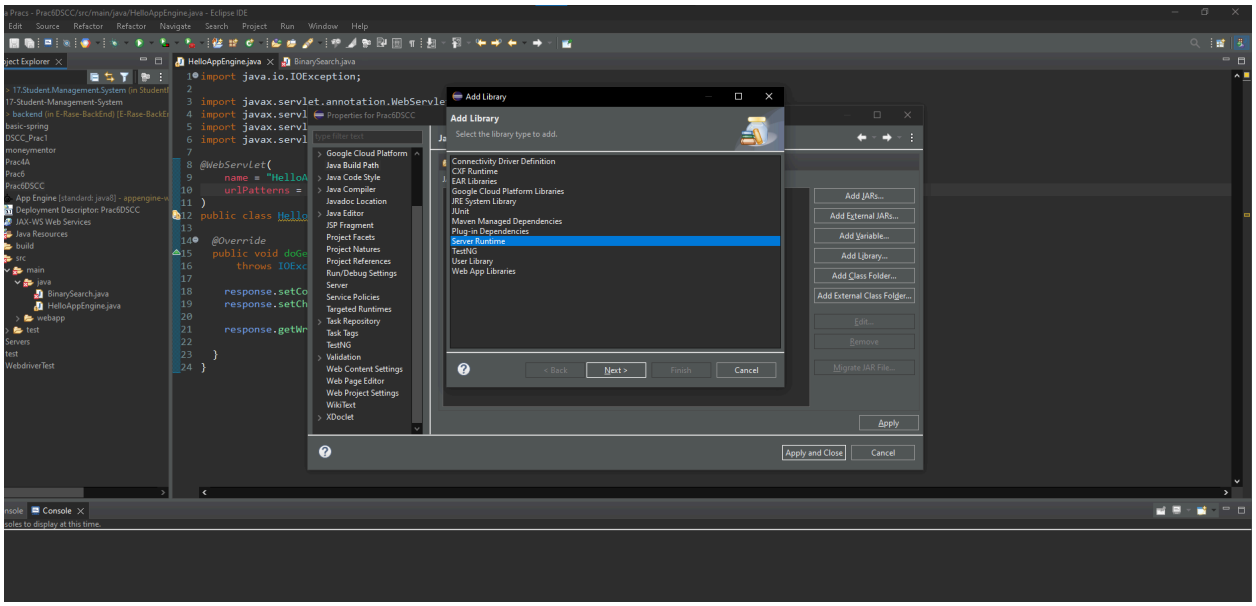
- Click Next, provide a project name (e.g., BinarySearchApp), and configure settings as required.

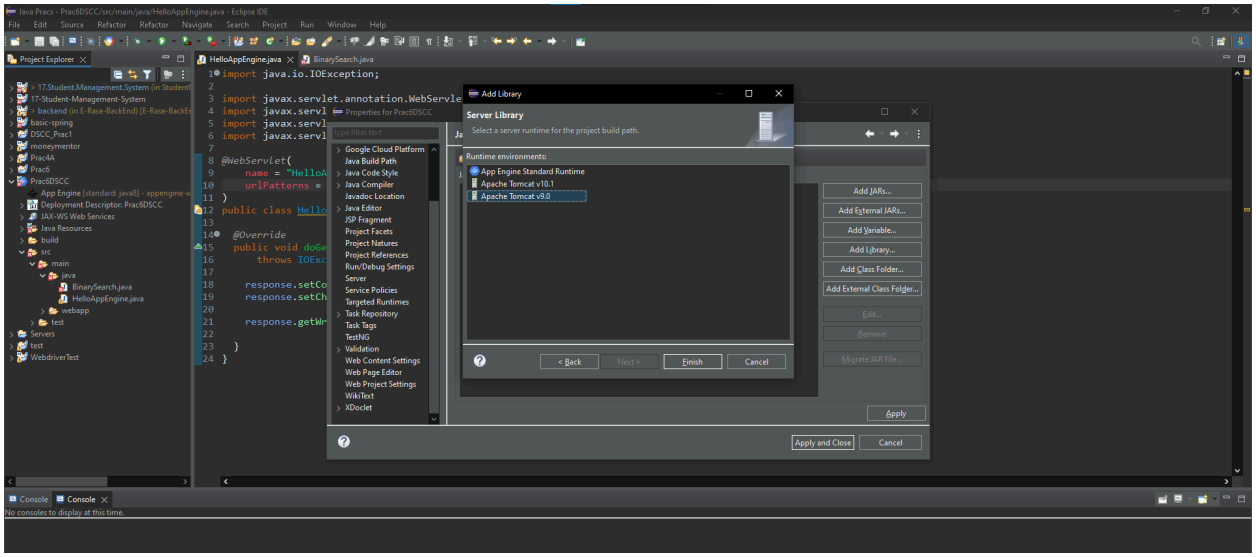
Add Servlet API to Your Project:

- Right-click on your project in the Project Explorer in Eclipse.
- Select **Properties**.
- Go to **Java Build Path** and then click on the **Libraries** tab.



- Click **Add Library**.
- Choose **Server Runtime** (or search for **Servlet API** if available). Choose Tomcat v9.0 and click on Finish





Step 2: Implement the Binary Search Algorithm

1. **Create a new Java class**:
    - Right-click on src/main/java and select New -> Class.
    - Name the class (e.g., BinarySearch).
2. **Implement the binary search method**: Here's a simple implementation of the binary search algorithm:

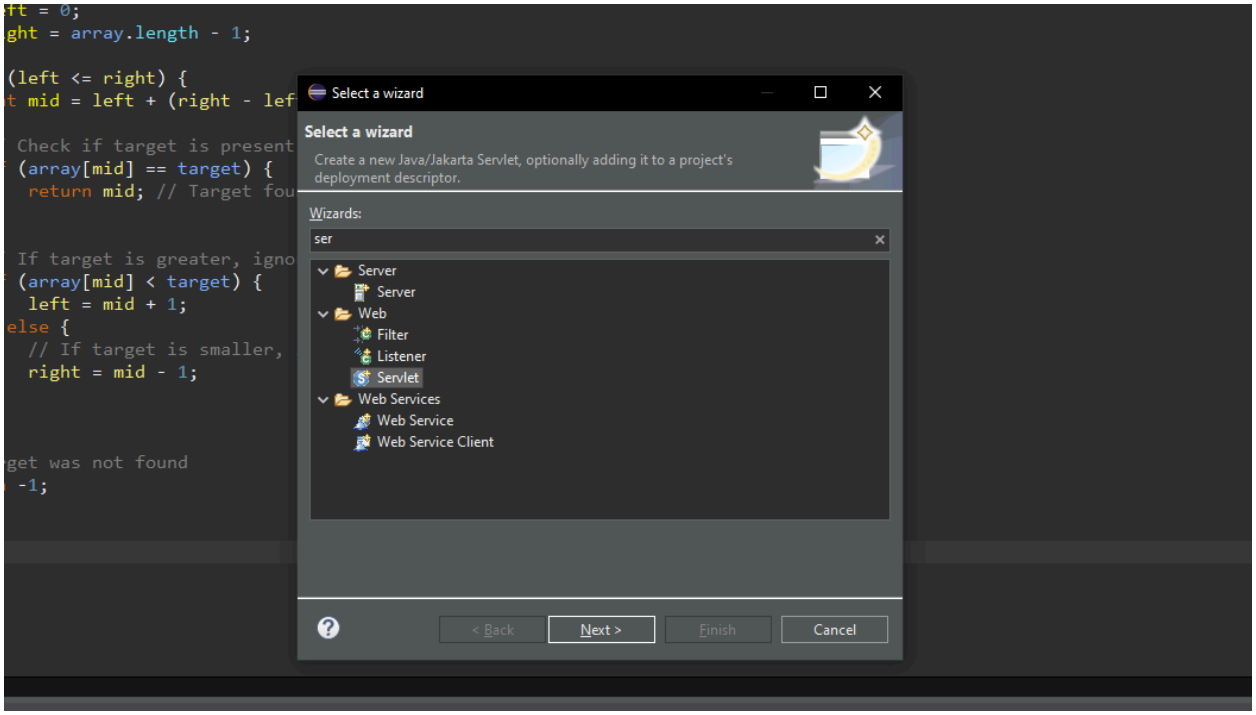
BinarySearch.java

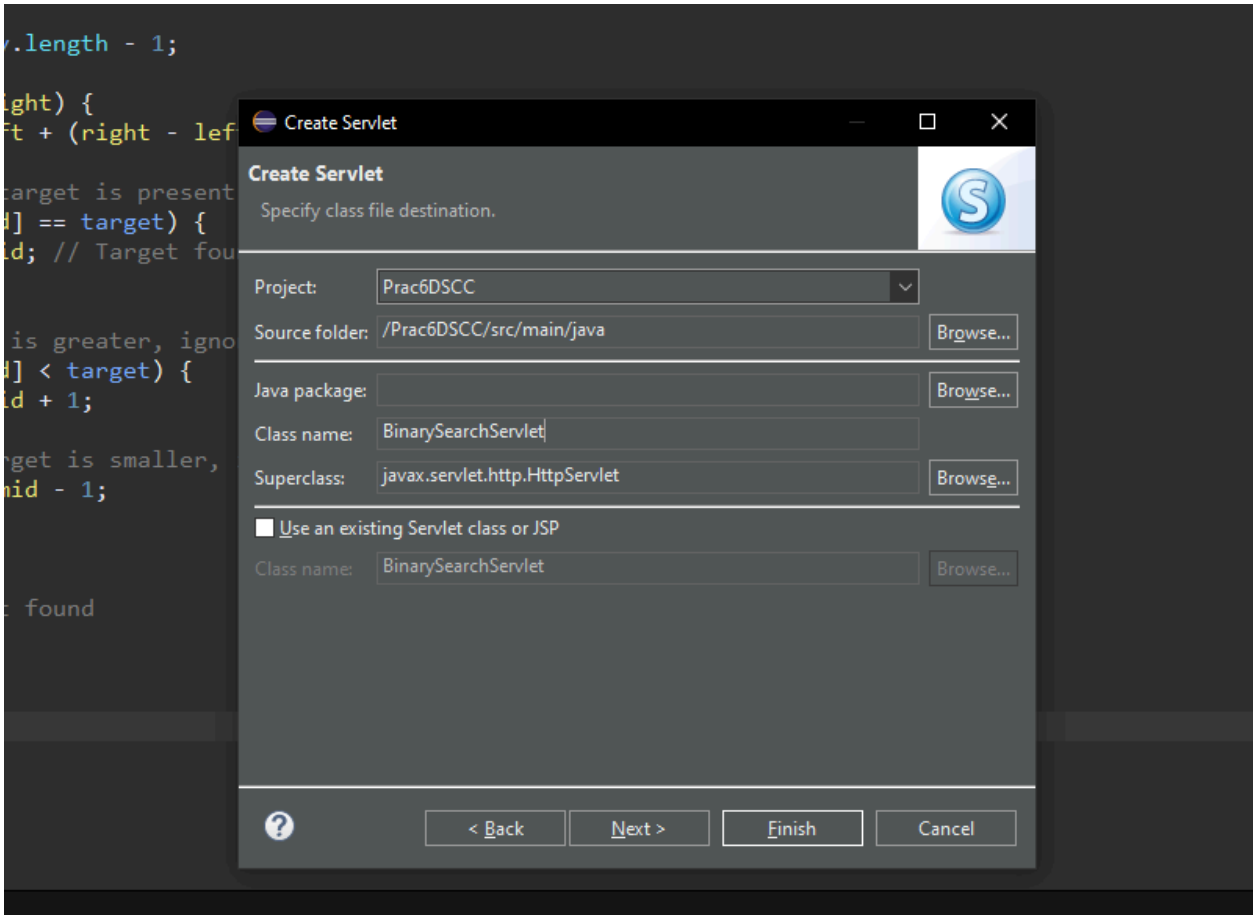
```
public class BinarySearch {

    public static int binarySearch(int[] array, int target) {
        int left = 0;
        int right = array.length - 1;

        while (left <= right) {
            int mid = left + (right - left) / 2;

            // Check if target is present at mid
            if (array[mid] == target) {
```

```
            return mid; // Target found at index mid
        }

        // If target is greater, ignore left half
        if (array[mid] < target) {
            left = mid + 1;
        } else {
            // If target is smaller, ignore right half
            right = mid - 1;
        }
    }

    // Target was not found
    return -1;
  }
}
```

Step 3: Create a Servlet to Handle Requests

1. **Create a new Servlet**:
    - Right-click on src/main/java and select New -> Servlet.
    - Name the servlet (e.g., BinarySearchServlet).


ft = 0;
ght = array.length - 1;
(left <= right) {
t mid = left + (right - lef
Check if target is present
(array[mid] == target) {
return mid; // Target fou
If target is greater, igno
(array[mid] < target) {
left = mid + 1;
else {
// If target is smaller,
right = mid - 1;
get was not found
-1;
Select a wizard
Select a wizard
Create a new Java/Jakarta Servlet, optionally adding it to a project's deployment descriptor.
Wizards:
ser
Server
Server
Web
Filter
Listener
Servlet
Web Services
Web Service
Web Service Client
< Back
Next >
Finish
Cancel



2. **Implement the servlet to handle HTTP requests**: Here's an example of how to set up the servlet:

BinarySearchServlet.java

```
import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet("/search")
public class BinarySearchServlet extends HttpServlet {
```

```
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {

        // Get parameters
        String[] arrayStr = request.getParameterValues("array");
        int target = Integer.parseInt(request.getParameter("target"));

        // Convert String array to int array
        int[] array = new int[arrayStr.length];
        for (int i = 0; i < arrayStr.length; i++) {
            array[i] = Integer.parseInt(arrayStr[i]);
        }

        // Perform binary search
        int result = BinarySearch.binarySearch(array, target);

        // Set the result in the response
        response.setContentType("text/plain");
        if (result != -1) {
            response.getWriter().println("Target found at index: " + result);
        } else {
            response.getWriter().println("Target not found.");
        }
    }
}
```

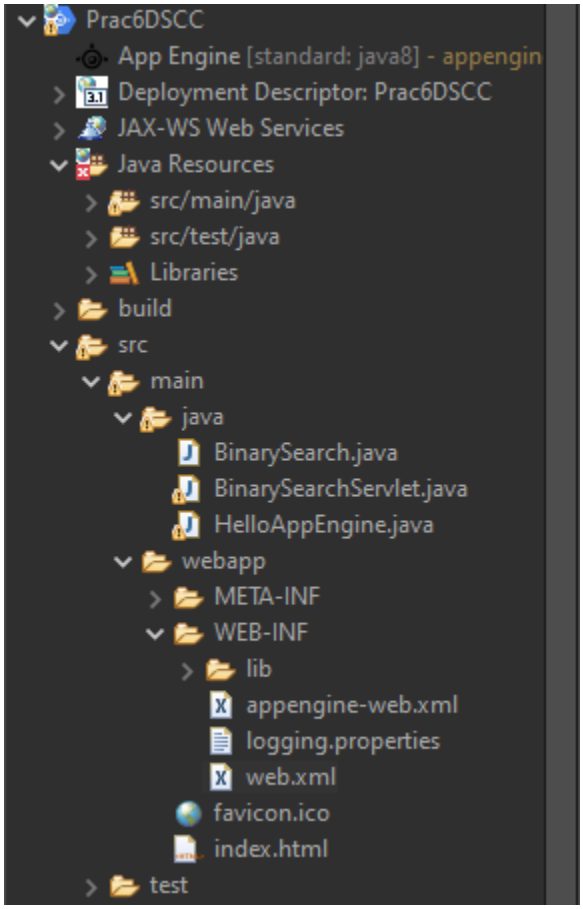
Step 4: Deploy the Application

1. **Configure Deployment Descriptor**: Make sure to define your servlet in the web.xml file under src/main/webapp/WEB-INF/.

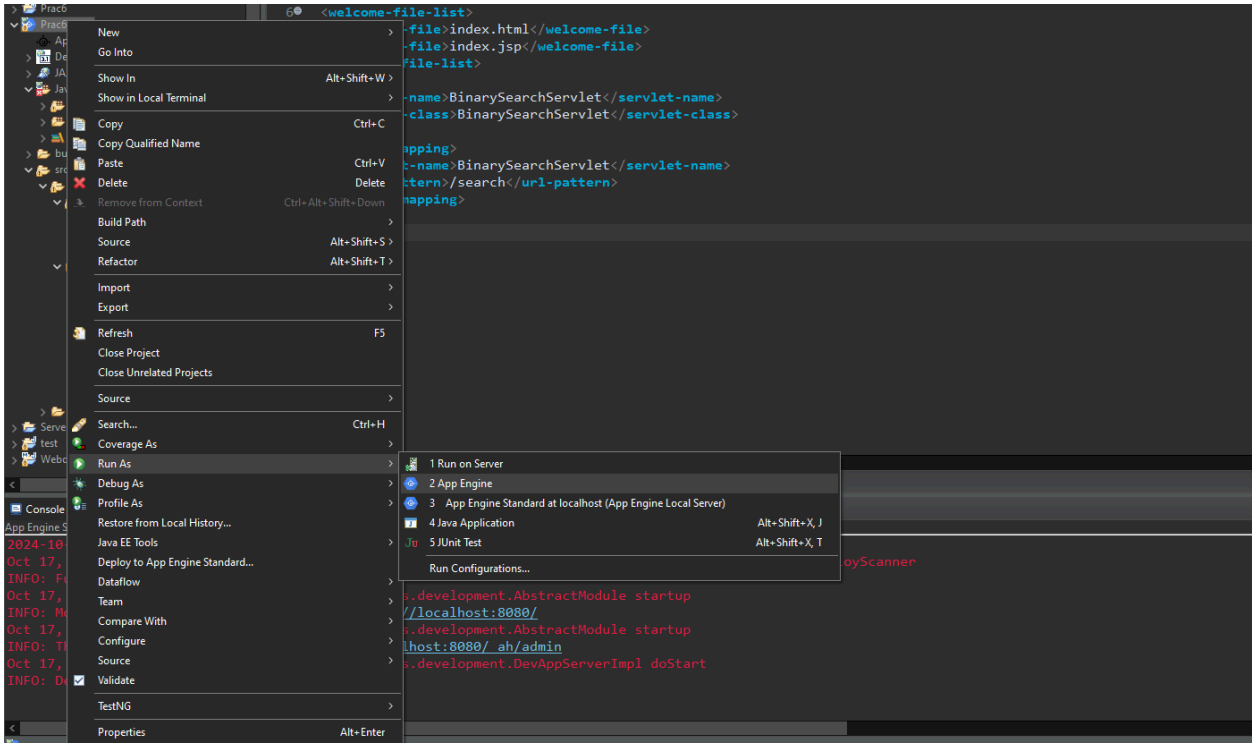
web.xml

```
<?xml version="1.0" encoding="utf-8"?>
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                                                xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
        version="3.1">
  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
    <welcome-file>index.jsp</welcome-file>
  </welcome-file-list>
  <servlet>
    <servlet-name>BinarySearchServlet</servlet-name>
    <servlet-class>your.package.name.BinarySearchServlet</servlet-class>
  </servlet>
  <servlet-mapping>
          <servlet-name>BinarySearchServlet</servlet-name>
          <url-pattern>/search</url-pattern>
  </servlet-mapping>
</web-app>
```

Folder Structure

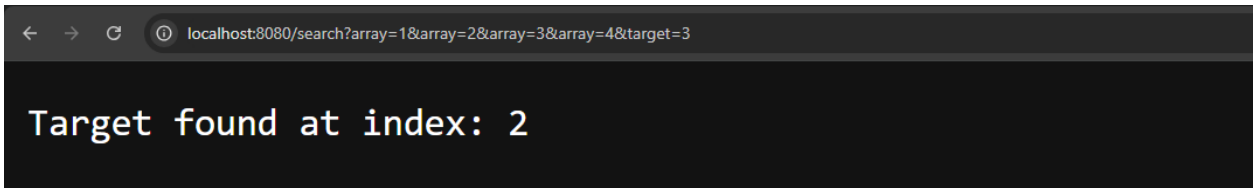


2. **Run the application**:
   - Right-click on your project in the Project Explorer.
   - Select **Run as > App Engine**.



Once the application start running, On browser URL, search for

http://localhost:8080/search?array=1&array=2&array=3&array=4&target=3



Conclusion:

In this practical, we successfully developed a web application using Google App Engine in Eclipse IDE that implements a binary search algorithm to find the position of a target value within a sorted integer array. By setting up the Google Cloud Tools plugin, writing the binary search logic, and configuring servlets to handle HTTP requests, we demonstrated how to efficiently search through data using a fundamental algorithm in a cloud environment. Additionally, resolving issues such as missing dependencies (e.g., Servlet API) further reinforced the importance of proper project configuration when working with web applications.