GIS TOOL FOR HELLENISTIC KINGDOMS

A Thesis
Presented to the
Faculty of
San Diego State University

In Partial Fulfillment
of the Requirements for the Degree
Master of Science
in
Computer Science

by
Varun Penumalla
Fall 2015
SAN DIEGO STATE UNIVERSITY

The Undersigned Faculty Committee Approves the

Thesis of Varun Penumalla:

GIS Tool for Hellenistic Kingdoms

Carl Eckberg, Chair
Department of Computer Science

Alan Riggins
Department of Computer Science

Peter Blomgren
Department of Mathematics and Statistics

Oct 30, 2015
Approval Date
DEDICATION

I would like to dedicate this thesis to my mother, father, brother and friends who always supported, encouraged and stood by me for accomplishing my work.
ABSTRACT OF THE THESIS

GIS Tool for Hellenistic Kingdoms
by
Varun Penumalla
Master of Science in Computer Science
San Diego State University, 2015

The aim of this thesis is to develop a tool which depicts information about the Hellenistic Empires in an interactive way. The tool covers major events involving different dynasties starting from the rise to the fall of the Hellenistic empire which covers from 323 BC to 31 BC. This goal is achieved by building a GIS application in combination with an interactive website which displays information about the kingdoms and their pivotal battles.

The GIS Application developed using MOJO is a GUI tool which depicts information about the kingdoms which flourished in Hellenistic Civilization. The information is represented using a map which provides the user with details about the kingdom’s boundaries and battles that occurred in a specific region between different empires. Two kinds of layers are employed to showcase this information. Shapefiles from .csv files are created for a specific kingdom. These .csv files contain the co-ordinates of a specific region where the battles have occurred. A custom tool is used to display information about the battle location and a link to the website in a separate window.

The website is built using HTML and CSS and is further made responsive using various front end technologies like Bootstrap and Angular JS. The website is also made dynamic using server side scripting languages like Node.JS and a back-end database like MongoDB.

A Timeline feature containing information about the kingdoms and battles that have occurred over three centuries, under different rulers, is developed using JavaScript and Bootstrap. The Website is hosted on a local server and the data is displayed in the form smart tables.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>SECTION</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABSTRACT</td>
<td>v</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>ix</td>
</tr>
<tr>
<td>ACKNOWLEDGEMENTS</td>
<td>xi</td>
</tr>
<tr>
<td>CHAPTER</td>
<td></td>
</tr>
<tr>
<td>1 INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Overview</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Motivation</td>
<td>1</td>
</tr>
<tr>
<td>2 TECHNOLOGIES USED IN THE APPLICATION</td>
<td>3</td>
</tr>
<tr>
<td>2.1 JAVA</td>
<td>3</td>
</tr>
<tr>
<td>2.2 Mapobjects Java Edition</td>
<td>4</td>
</tr>
<tr>
<td>2.3 Web Technologies</td>
<td>5</td>
</tr>
<tr>
<td>2.4 NODE JS</td>
<td>5</td>
</tr>
<tr>
<td>2.5 Express Web Framework</td>
<td>5</td>
</tr>
<tr>
<td>2.6 Mongo DB</td>
<td>6</td>
</tr>
<tr>
<td>2.7 Angular JS</td>
<td>6</td>
</tr>
<tr>
<td>2.8 Bootstrap</td>
<td>6</td>
</tr>
<tr>
<td>3 REQUIREMENTS</td>
<td>7</td>
</tr>
<tr>
<td>3.1 Data Requirements</td>
<td>7</td>
</tr>
<tr>
<td>3.2 Platform Requirements</td>
<td>7</td>
</tr>
<tr>
<td>3.3 Functional Requirments</td>
<td>8</td>
</tr>
<tr>
<td>4 ENVIRONMENT SETUP</td>
<td>9</td>
</tr>
<tr>
<td>5 SOFTWARE IMPLEMENTATION</td>
<td>12</td>
</tr>
<tr>
<td>5.1 Launching the Application</td>
<td>12</td>
</tr>
<tr>
<td>5.2 Zoom Pan Toolbar</td>
<td>13</td>
</tr>
<tr>
<td>5.3 Selection Toolbar</td>
<td>13</td>
</tr>
</tbody>
</table>
5.3.1 Search Tool ................................................................. 14
5.3.2 Find Tool ................................................................. 14
5.3.3 Select Tool ............................................................... 14
5.3.4 Query Builder .......................................................... 15
5.3.5 Clear All Selections ............................................... 15
5.3.6 Buffer ...................................................................... 15
5.3.7 Attributes .................................................................. 16
5.4 Custom Toolbar .......................................................... 16
5.4.1 Print Tool ................................................................. 16
5.4.2 Add Layer ................................................................. 17
5.4.3 Pointer Tool .............................................................. 17
5.4.4 Distance Measuring Tool ........................................ 17
5.4.5 Hotlink Tool .............................................................. 18
5.4.6 XY Tool .................................................................... 18
5.4.7 Help Tool .................................................................. 19
5.5 Menu Bar ................................................................. 20
5.5.1 File Menu ................................................................. 20
5.5.2 Theme Menu ............................................................ 21
5.5.3 Layer Control Menu .............................................. 22
5.5.4 Help Menu ............................................................... 23
5.5.5 Dynasties Menu ..................................................... 23
6 WEBSITE IMPLEMENTATION ........................................... 25
6.1 Directory Structure ..................................................... 26
6.2 Configuring the Server File ....................................... 27
6.3 Configuring Database ................................................. 28
6.4 Client Side Controllers .............................................. 30
6.5 Querying and Saving Data ......................................... 30
6.6 Filters ......................................................................... 31
6.7 Dynamic Generation of Page ...................................... 32
7 OVERVIEW OF THE PROJECT ........................................ 33
8 SUMMARY ...................................................................... 39
8.1 Difficulties .................................................................. 39
8.2 Future Scope ..............................................................40
REFERENCES .................................................................41
LIST OF FIGURES

PAGE

Figure 4.1. Java eclipse project.................................................................9
Figure 4.2. Compiling java files through command line................................10
Figure 4.3. Running java application using the command moj_run.................10
Figure 4.4. Initiating the database daemon process through mongod...............10
Figure 4.5. Starting the server through the command line.............................11
Figure 4.6. Loading the homepage at 3000 port...........................................11
Figure 4.7. Mongo client........................................................................11
Figure 5.1. Application launch page..........................................................12
Figure 5.2. Zoom pan toolbar.................................................................13
Figure 5.3. Selection toolbar....................................................................14
Figure 5.4. Find tool window.....................................................................14
Figure 5.5. Selection feature shapes............................................................14
Figure 5.6. Query builder.........................................................................15
Figure 5.7. Buffer window........................................................................15
Figure 5.8. Attributes table.......................................................................16
Figure 5.9. Custom toolbar.......................................................................16
Figure 5.10. Print dialogue box.................................................................16
Figure 5.11. Shapefile selection toolbar......................................................17
Figure 5.12. Distance measuring tool.........................................................18
Figure 5.13. Hotlink tool..........................................................................18
Figure 5.14. CSV file................................................................................19
Figure 5.15. CSV file selection.................................................................19
Figure 5.16. File menu..............................................................................20
Figure 5.17. Legend editor........................................................................20
Figure 5.18. Theme menu........................................................................21
ACKNOWLEDGEMENTS

I would like to thank my advisor and Thesis chairperson Professor Dr. Carl Eckberg for giving me an opportunity to work on this thesis and for the continuous support of my study and research.

Also, I would like to thank Professor Alan Riggins, Computer Science Department and Professor Peter Blomgren, Mathematics and Statistics Department of San Diego State University for being on my thesis committee and co-operation throughout.
CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

A Geographic Information system is a combination of Information system and spatial data. A GIS application allows us to query spatial data from a database which can be visualized and analyzed in the form of a map. Geographical data which consists of co-ordinates of locations are projected onto map in any GIS application [1].

The aim of the thesis is to build an interactive learning GIS Application coupled with a dynamic Website to give the end user a better understanding of kingdoms in the Hellenistic Period and Battles that occurred between them. The java module comprises of tools and layers containing data points which focus on where the battles occurred and the boundaries of the kingdoms. The Web module consists of information of the battles displayed in an interactive way.

Map Objects Java Edition is a product released to developers to build Java Applications by ESRI which produces software to build of Geographic Information System software and geodatabase management applications. Another part of thesis is to build a MEAN stack website which co-exist with the GIS application. The user can access information about the rulers, dynasties and battles in an interactive way possible.

The chapters in this thesis document provides information about the technologies used, runtime setup used, data requirements, software implementation of Java application and Website implementations and in the last an overall view of the project.

1.2 MOTIVATION

Hellenistic civilizations located in Greece or Macedonia represents the influence of Greek from 323 BC – 31 BC [2, 3]. These civilizations are preceded by Hellenic period and followed by Roman Empire [4, 5]. This thesis is intended for studying about the dynasties
emerged in this era, events and wars which occurred after the death of Alexander the great [6, 7, 8].
CHAPTER 2

TECHNOLOGIES USED IN THE APPLICATION

This thesis is an integration of Java technologies, especially Map Objects Java Edition, with an Interactive Web Application. Java is platform Independent and there are a number of packages to support the development of these desktop applications. One of the ways to display the Geographical Information in the Java Environment is by using MOJO, which is a collection of packages provided by ESRI. The Interactive Website is built using the present trending technologies such as the MEAN Stack which are known to complement well with each other.

2.1 JAVA

Java is a high level programming language and a computing platform developed by Sun Microsystems. A computing platform is a software or a hardware environment in which an application runs. JRE is the run time environment of Java and it contains the JVM which is required for running the program. JVM executes the bytecode generated from Java source code by using a Java compiler. Bytecodes are represented by 1’s and 0’s and are stored in class files. A java program can be compiled on any operating system since it is a platform independent language and hence the tagline Write Once Run Anywhere [9].

The features which are considered to be the core strengths of the language are listed below:

**Object Oriented:** Object-oriented refers to Software organization as a combination of different types of objects that incorporates both data and behavior. Main concepts include Inheritance, Polymorphism, Abstraction and Encapsulation.

**Simple:** Its syntax is based on C++ and has features like garbage collection and developing application devoid of pointers

**Secure:** Has no explicit pointers and runs on Bytecode generated by the compiler.

**Portable:** The bytecode can be generated on any platform.
Robust: Java has strong memory management along with exception handling and type checking mechanisms which makes it robust [10].

2.2 MapObjects Java Edition

MapObjects Java Edition is collection of packages written by ESRI to create Dynamic Maps which can be integrated into Java development environment. The API required for building Java based Geo-content applications is a library consisting of java classes, distributed in the form of a jar files. It provides a number of classes which are used for analyzing, manipulating geographical data and querying on its spatial Information without requiring any other middleware software [11].

Map Objects library files are written in Java. This, along with the Java AWT and Swing libraries (Graphical User Interface Toolkits of Java) provide the developer with an appropriate interface to focus on developing a customized geographic Java based application.

The existing Shape files and relevant files from the internet and intranet can be combined for creating custom maps. Also, these Shapefiles along with ArcSDE layers, image formats such as BMP, TIFF, PNG, JPEG, and GIF can be integrated into the application, providing a wide range of GIS capabilities at the same time [12].

MapObjects –Java Edition supports Client and Server Side GIS applications. Client side components allows the developer to create a User Interface, query and data retrieval activities while Server side components allows the developer to build map services, JSPs, EJBs and use industry standard J2EE application server providers [12].

MOJO provides predefined toolbars such as Zoom Pan tool bar for viewing the graphical data in a map and Selection tool bar for selecting a region of a map. With the aid of Java GUI toolkit, custom toolbars are also integrated into the application. Use of a legend to select a specific layer and retrieve its information also makes the application simple and effective [13].

MOJO provides classes for creating the custom layers, promoting/demoting the layers and displaying its relevant information in a Legend, according to the priority of the layer selected. It also features constructing point data on a layer from a comma separated value file. It gives programmers enough space to build user defined tools like calculating the distance from one point to another taking into account their latitude and longitude by using various projection coordinate systems.
2.3 WEB TECHNOLOGIES

A dynamic website can be described as a group of web pages where the data is dynamically generated or changed with time along with the data the user accesses upon. Client and Server side scripting form the main architecture of the Dynamic Website. Modern day technologies such as the MEAN (MongoDB, Express, Angular.js, Node.js) stack, which overshadow previous technologies like LAMP, are used to build the current Website which showcases relevant information of the project. Node.js and Express Web Framework form the Server side Architecture and Angular.js is introduced for the client side scripting along with Bootstrap as the Front End Framework. MONGODB is a No-SQL database which is used for backend transactions [14].

2.4 NODE.JS

Node.js is a JavaScript based framework built on Google Chrome’s JavaScript V8 Engine which is generally used to develop Server side Web Applications. Node.js is JavaScript, running on Server side. It is an open source software designed for building scalable applications like web servers such as the Apache HTTP server. Its libraries are asynchronous which means the program continues to perform the rest of the tasks while waiting for a response by registering callbacks [15].

A multi-threaded approach is generally a perfect way for concurrency in applications but has some drawbacks like deadlocks and unavailability to use shared resources. Nodejs overcomes these drawbacks since it runs on a single thread and is based on Event driven model, which makes the application lightweight and efficient.

Node JS has a pre-installed package manager called NPM (Node Package Manager) which is used to install programs or packages and its dependencies. It also provides command line utility and installs the packages locally or globally depending on the requirement. Also, different versions of modules can be handled by the NPM.

2.5 EXPRESS WEB FRAMEWORK

Express.js is a Node.js web application server framework, designed for building hybrid web applications. It acts as a middleware software which basically accepts the request and generates the response accordingly. Server side routing is handled by ExpressJS by using
HTTP request methods GET and POST. Responses are sent back to the browser in the form of JSON data or Webpages. Also, ExpressJS loads files under a specific user defined directory for easy access. Authentication from the database, REST API calls and Error handlers are employed in the server configuration file using Express framework [16].

2.6 MONGO DB

Mongo DB can be viewed as a NoSQL document oriented database, a non-relational database or as a semi structured database written in C++. It is an open source software and the fourth most popular database in the present times. Data is stored in JSON format and each row of data is called document. All the data is stored in a single document rather than the conventional structuring of data into different tables. A set of documents are stored in a collection which is similar to a table of data. Another advantage of mongo DB is its flexibility in schema design where documents in same collection can have different fields. It generates a unique id for each and every distinctive document. MongoDB is used for developing applications which are scalable and performance oriented and sits well with Node.JS [17].

2.7 ANGULAR JS

Angular JS is a powerful JavaScript framework to build large scale and high performance applications. It provides the developer with the options to build dynamic client side web apps and single page applications in a clean MVC pattern. AngularJS components combined with HTML, provide the user with a responsive web application. It also uses Dependency Injection and Data binding in client side environment. Much of the application logic is written in the controller which is a JavaScript file and is linked to a HTML page as its view. A model contains the data which can be accessed from the controllers and responds to the request made by the user from the view [18].

2.8 BOOTSTRAP

Bootstrap is a front end framework used to make the user Interface of the website simpler and cleaner. It is basically a CSS framework consisting of pre-defined classes and attributes which makes the website more responsive. Usage of this framework saves time by building web applications faster with ease.
CHAPTER 3

REQUIREMENTS

The aim of the tool which is a combination of a GIS Application and the MEAN stack Website is to showcase the user with detailed information about the Hellenistic Kingdoms and the battles that took place between them from 323 BC – 31 BC. This thesis has been developed under the guidance of Dr. Carl Eckberg from San Diego State University, Computer Science Department. The requirements are divided into three categories as follows: Data requirements, Platform requirements, Functional requirements.

3.1 DATA REQUIREMENTS

The data gathered to showcase the details about different kingdoms, which lasted for over three centuries, required a visiting to many historical and educational websites. There are mainly 2 types of layers in the GIS Application developed. The kingdom layers where the information related to the kingdom areas are spread out. The Battle layers where the battles have occurred. The latter one is displayed using the hotlink tool. Also, the application is equipped with data related to the battles and hypertext links which redirect the user to the website.

3.2 PLATFORM REQUIREMENTS

All the MOJO files can be run from the command line where java is installed. User friendly environment such as the Eclipse IDE is used since all the code is written in Java. Sublime editor is used for developing the web application which mostly consists of JavaScript libraries on client and server side. The different technologies used for developing this interactive tool are as follows:

- JAVA
- MapObjects – Java Edition
3.3 Functional Requirements

Functional requirements gives an outline of how a project should work and details about its features and functionalities. It refers to the specifics of how the tool should be functioning and the User Interface. The requirements for making this interactive tool are listed below:

- The shapefiles are created for each and every kingdom and a default base map is displayed when the application is launched.
- The app must contain tools like zoom and pan in and pan out.
- A custom toolbar containing Hotlink to get information about the battles and Print button.
- Tools to add and Remove layer dynamically.
- A help menu describing how a tool or a feature works.
- .CSV files containing information about battle places.
- The application must contain Table of Contents where all the layers are displayed and the contents can be edited.
- The final application must be a jar file where all the classes are compiled, for easy deployment
- Website containing all the information of Rulers and specific Dynasties.
- Battle Information which is dynamically generated from the database.
- A user selecting a range of years for rulers must display all the rulers in that range for any kingdom.
CHAPTER 4

ENVIRONMENT SETUP

The GIS application can be started using the command line or an IDE. The path for MOJO and Java must be set first. Eclipse IDE was used for writing the Java classes and MOJO jar files are imported as external libraries by selection configuring build path in options menu. The application is started by clicking on the main method class and then running it as a Java application.

Figure 4.1. Java eclipse project.

These java files can also be compiled from the command line using the command moj_compile and then running the application by using moj_run followed by the required classes. To convert these class files into a jar file, a manifest file is created which describes the location of class files and specifics of library files.
Figure 4.2. Compiling java files through command line.

```
C:\Users\Home\Desktop\ThesisProject\Source\moj_compile *.java
Note: Some input files use or override a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
Note: Some input files use unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
Press any key to continue . . .
```

Figure 4.3. Running java application using the command moj_run.

MongoDB server is started which responds to the database requests on 27017 port. The location where the database files are stored is specified. Having started the Database server, the user can now be redirected to the website from the JAVA – MOJO application.

```
C:\\mongod --dbpath C:\Users\Home\Documents\data
2015-10-01 03:23:51,425 I JOURNAL [initandlisten] journal dir=C:\Users\Home\Documents\data\journal
2015-10-01 03:23:51,622 I CONTROL [initandlisten] MongoDB starting : pid=24524 port=27017 dbpath=C:\Users\Home\Documents\data\data
2015-10-01 03:23:51,622 I CONTROL [initandlisten] db version v3.0.4
2015-10-01 03:23:51,622 I CONTROL [initandlisten] Build info: windows sys.getwindowsversion(major=6, minor=1, build=7600)
2015-10-01 03:23:51,622 I CONTROL [initandlisten] options: { storage: { dbPath: "C:\Users\Home\Documents\data" } }
2015-10-01 03:23:51,775 I NETWORK [initandlisten] waiting for connections on port 27017
```

Figure 4.4. Initiating the database daemon process through mongod.
The Nodejs server is started by invoking its server.js file from the command line, by typing “node server.js”. A port is then opened for deploying the application on a specific port specified in the program.

The Web server starts running and the web pages can be accessed by the user.

The mongo client can be started by typing “mongo” in the command line. The database path is also specified along with the server specification, which makes the client automatically connect to the location and wait until further instructions.

![Command Prompt - nodemon server.js](image)

**Figure 4.5.** Starting the server through the command line.

![Hellenistic Kingdoms](image)

**Figure 4.6.** Loading the homepage at 3000 port.

![Command Prompt - mongo](image)

**Figure 4.7.** Mongo client.
CHAPTER 5

SOFTWARE IMPLEMENTATION

This chapter involves software implementation along with its detailed explanation. Screenshots are presented, wherever necessary, to show the user how a particular tool functions.

5.1 LAUNCHING THE APPLICATION

As soon as we run the GIS application, which is converged into a jar file, it opens the main page of the application which displays a base map along with the table of contents, status bar, tool bars and legends, which display the name of every layer. These topics are covered in depth in this chapter at a later stage.

![Application launch page](image)

Figure 5.1. Application launch page.

MOJO provides two predefined toolbars which can be used with the GIS application. These toolbars have different functionalities that help the user get his required actions performed, when the application is running. The two built-in toolbars are Zoom pan Toolbar and Selection Toolbar. The Zoom pan toolbar contains tools which help the user change the
view of the map. Selection Tool bar is used to select some features from the map and retrieve information from the selected layers. Along with these, a custom toolbar is also built to give the user more features along with the existing ones. A detailed explanation of how the tool works is discussed below.

5.2 ZOOM PAN TOOLBAR

A list of tools which are present in ESRI tool bar along with their functionalities are discussed below. The figure below depicts the Zoom pan tool bar.

![Figure 5.2. Zoom pan toolbar.](image)

The above tool bar comes with nine standard buttons and each of them has a unique functionality to manipulate the behavior of the existing map in the applications. The tools in the Zoom pan toolbar are the following:

- Zoom to Full Extent – Zooms to the entire area of the map
- Zoom to Previous Extent – Zooms to the previous display of the map
- Zoom to Next Extent – Zoom to the next display of the map
- Zoom to Active Layer – Zooms the map to all the selected regions
- Zoom In – Zooms into the region selected on the map
- Zoom Out – Zoom out of the current region
- Pan – Drag the map in the direction of the mouse
- Pan in one direction – Drag the map in a particular direction
- Identity – Tells specific information about the selected feature in the map.

5.3 SELECTION TOOLBAR

The below ESRI toolbar depicts Selection Toolbar gives the user, information about the data of the selected area or a layer. The toolbar comes with a standard 7 buttons and each of their functionalities is described below individually.
5.3.1 Search Tool

This tool is used to display results from a predefined stored query.

5.3.2 Find Tool

When user selects this tool, a dialog box opens with a list of layers and options like the Pan and Zoom. A particular place in a map is displayed depending on the value entered. If the entered value, corresponding to a place, is present in the map layer, it gets highlighted in the map and the options like Pan and Zoom are enabled to work on.

Figure 5.4. Find tool window.

5.3.3 Select Tool

This tool allows the user to highlight a section in a map, which can be later saved to create a new shape file. The features in the map can be highlighted using a line, circle, rectangle or any polygon shape.

Figure 5.5. Selection feature shapes.
5.3.4 Query Builder

When this tool is selected, a dialog box appears where we can retrieve information related to the selected layer. Information is retrieved by executing SQL type queries. A query is executed so as to match user expected results and are then displayed in the bottom dialogue box. Upon selecting the result, Options of zoom, Pan, saving and highlighting the section of the map are enabled.

![Query builder](image)

**Figure 5.6. Query builder.**

5.3.5 Clear All Selections

Removes the selected features from the layer and displays the default selected layer.

5.3.6 Buffer

A buffer is used to form a region covering the selected feature layer on a particular map when buffer distance is given as its input.

![Buffer window](image)

**Figure 5.7. Buffer window.**
5.3.7 Attributes

This tool, when selected, gives information about the attributes of a particular feature. A dialogue box with table describing the data of a particular section appears as a result.

![Figure 5.8. Attributes table.](image)

5.4 CUSTOM TOOLBAR

A custom tool bar is incorporated into the application. It gives the end user, extra features like adding a layer by using csv files. Having such a feature replaces the conventional way of loading stored maps, shapefiles at the time of application launch. The toolbar is designed with following functionalities which are represented by the image below.

![Figure 5.9. Custom toolbar.](image)

5.4.1 Print Tool

This tool is used to print the existing map or save it in the form of a PDF. The options can be selected from the following dialogue box.

![Figure 5.10. Print dialogue box.](image)
5.4.2 Add Layer

This tool facilitates the user to add a shapefile to the existing layers. A dialogue box appears from which we can select the location path of the shapefile, image file or one of the specified types of layers that ESRI allows. The next step is to click on OK button. The layer then gets added to the top of the Table of contents.

![Shapefile selection toolbar](image)

Figure 5.11. Shapefile selection toolbar.

5.4.3 Pointer Tool

This tool is used to change/remove the behavior of the selected tool to a default behavior. When any other tool is selected, the mouse pointer symbol changes and accordingly its behavior. When this tool is selected, the behavior is reverted to the default and the cursor symbol is changed as an arrow icon.

5.4.4 Distance Measuring Tool

This tool provides the user to measure the distance between two points in the map. To measure the distance using the mouse, initially a point is clicked and then is dragged until the desired point and is released. A line automatically appears between the two points and the resultant distance between them is displayed in the status bar in miles as well as in kilometers.
5.4.5 Hotlink Tool

The main aim of the Hotlink tool is to select a point from the data layer and display its data in our project. The below image represents information of a selected point on a map. A dialog box pops up which lists the battles that occurred at the selected point and displays the information about the event and its appropriate web links to which a user can be redirected. The underlying functionality can be described as “When a points layer is selected, this hotlink tool searches for the corresponding CSV file and if there is a match, the Window (dialog box) appears.

![Figure 5.12. Distance measuring tool.](image)

5.4.6 XY Tool

A tool which is used to dynamically generate a layer from the CSV file. The CSV file is a list of latitudes and longitudes with corresponding battle names in it. Once the file is selected, a layer with those co-ordinates is formed and added into the Table of Contents (TOC). The points are cast on the map as below.

![Figure 5.13. Hotlink tool.](image)
The below figure displays a dialogue box from which a CSV file is selected, pointing to particular places in a map. The details of the imported points are displayed along with its Layer name in the Table of Contents.

**5.4.7 Help Tool**

This tool helps to display information about a particular tool by using the right click button. A dialogue box containing information about the selected tool gets displayed.
5.5 Menu Bar

The Application has 5 menus namely File, Theme, Layer Control, Help, Dynasties. All the menus are discussed in depth in this chapter. The Menus are created using Swing toolkit of Java API. Swings are light weight, efficient and well embedded into the project. A list of Menus and the functionality of its items are discussed below.

5.5.1 File Menu

The File menu consists of mainly four sub menu items. Their functionalities are mentioned below.

- **Add Layer**: A new layer is added into the Table of contents by specifying the location of the shapefile.
- **Remove Layer**: This Item is used to remove the selected layer from the TOC.
- **Print Menu Item**: An existing map can be printed by using this item
- **Legend Editor**: Using this menu item, the shape, style, size and color of the symbol representing the points in a layer can be altered or changed and applied to a specific layer.

![Figure 5.16. File menu.](image)

![Figure 5.17. Legend editor.](image)
5.5.2 Theme Menu

The theme menu consists of mainly two sub menu items which have two different actions to be performed when selected.

- **Open Attribute Table**: To know the attributes of a specific layer, a layer is first selected from the TOC followed by selecting Open Attribute Table option. A Window appears listing all the data which the layer has. The figure below depicts the list of attributes—the base layer of the application has.

![Figure 5.18. Theme menu.](image)

- **Create Layer From Selection**: Features are selected from the existing base map to save it as a new shapefile. Having the previous step done, this menu item can then be selected. The features are selected using Select features tool from Selection Toolbar. A dialogue box appears where a custom name can be given to the shapefile. Then the layer automatically adds into the Table of Contents. The selected features are highlighted in a specific color. The below figures show the dialogue box and the selected features.

![Figure 5.19. Attributes table.](image)

![Figure 5.20. Dialogue box for saving a shapefile.](image)
5.5.3 Layer Control Menu

This menu item has two options which are related to controlling the layer in the application. A layer must be selected and followed by promoting or demoting a layer in order to move the layer up or down as required.

- **Promote Selected Layer:** A layer can be moved up the order, unless it is the top most layer in the TOC
- **Demote Selected Layer:** A layer is moved down the order, unless it is the last layer in the TOC

![Layer control menu](image)

Figure 5.22. Layer control menu.

![Demoting and promoting layers in table of contents](image)

Figure 5.23. Demoting and promoting layers in table of contents.
5.5.4 Help Menu

A Help menu consists of items which describe how a tool works by displaying information in a dialogue box. The Menu contains a sub menu which has the information about the tools used in the application.

![Help menu](image)

**Figure 5.24. Help menu.**

- **Help Tool:** Selection of this tool is done by clicking the right mouse button when hovering it. We can then know information about the other tools used in the application.
- **About Project:** A window appears with the abstract of the thesis when this menu item is selected.
- **Contact us:** A window appears with the information of the candidate and the university he studies in.

5.5.5 Dynasties Menu

This Menu consists of the names of dynasties which flourished in the Hellenistic time period. By selecting the one of the menu item, relevant kingdoms shape file along with the battle layer is projected on to the map area. By selecting again, the layers get vanished from the Map.

![Dynasties menu](image)

**Figure 5.25. Dynasties menu.**
The below figure depicts when a menu item Seleucid is selected, the map with battle points layer get displayed within the application.

Figure 5.26. Only layers of the selected menu item are shown.
CHAPTER 6

WEBSITE IMPLEMENTATION

The second part of the thesis is to build a dynamic website to showcase the information about the kingdoms and their battles in an interactive way. MEAN stack technologies are selected to build the dynamic website. The user gets redirected from the hotlink window (By clicking more information button) to the Battle information page in the Website which lists the information of the battle.

There are no static pages displayed in this website, meaning the data gets generated from the database depending on the corresponding page selected. Node.JS acts as a web server and Express.JS is the web application framework used to handle the requests from the end user and respond with appropriate web pages. Angular.JS is the front end framework from which the requests are called and results are generated from the Mongo DB database in JSON format and hence the data can be manipulated.

Package.json is the file, where all the dependencies required for the file are listed. It is a .json file which acts as an entry point for creating a web app. By executing the command `npm install` in the command line, all the packages are installed into the node_modules folder. NPM is the node package manager used for installation. The following figure depicts the contents of the Package.JSON file.

![Figure 6.1. Package.json file.](image)
6.1 Directory Structure

The following figure represents the file structure to be followed and it is followed in all the MEAN Stack Applications.

- **Bower components**: All the front end framework library files are installed in this folder. Bower module installs the required modules from the command line.

- **Node modules**: All the modules, which are required at the server side, are installed in this directory.

- **Public Folder**: Client side related files like angular controllers and views are present in this folder.

- **Server Folder**: Configuration files from the database and server side JavaScript controllers are placed in this folder. Database Schema files are located in the model directory.

- **Package.json**: All the dependencies are present in this file and can be installed on any other Computer by installing this file.

- **Server.JS**: This file is the Core file of the website. All the requests are handled by this file.
6.2 CONFIGURING THE SERVER FILE

All the required modules are initiated at the beginning of the file. Connection to the database is established and using the Express framework required components path location is specified. The following snippet covers on which port the server is to be run.

```javascript
var server = app.listen(3000, function () {
  var host = server.address().address;
  var port = server.address().port;
  console.log('Web app listening at http://%s:%s', host, port);
});
```

Figure 6.3. Server listening on port 3000.

Nodemon is a module for running and restarting the server repeatedly when changes are made in files related to the server. The command required for starting the server is `nodemon server.js`. The server starts and a database connection is established. The default location is specified at ‘/’ location which translates to http://localhost: portnumber/. In our case the port number is 3000.

Figure 6.4. Initiating the server.

The main use of Express is handling the HTTP requests such as GET and POST. The express handles the request and response is sent either in JSON format (if the call is a REST API call) or in a HTML view (when a particular URL is clicked). The following figure depicts the information described above.
6.3 Configuring Database

The database model needs Mongoose.js to connect to the database. Mongoose is an object model packaging module for node. Mongoose connects to MongoDB from the server file. It allows the user to define a schema and initialize a collection. A collection emulates a table and Schema can be described as the fields present in the table.

The values are stored in the database using the mongoose models when the POST function is called. The server calls a function from the database model file. This function is exported as a module using Node, so that the server can be called when needed. The following snippet code depicts how a schema is created and initiated.

```javascript
var battlebSchema = mongoose.Schema({
  nameOfBattle: {type: String},
  place: {type: String},
  kingdom1: {type: String},
  kingdom2: {type: String},
  victory: {type: String},
  year: {type: String},
  description: {type: String}
});

var battleeventmodel = mongoose.model('battleeventmodel', battlebSchema);
```

The figure below shows how the values are inserted into the database by the user. When all the fields are filled in and “Add to the database” button is clicked on the client side, a function is called & the data is posted. The server gets the request and redirects to the DBmodel file where the function to store input data is called.

Figure 6.5. App GET/POST.
Another way to insert bulk data is to store data of events in a CSV file and by using `mongoimport` command. The database name, collection name and other details mentioned are executed in the mongoClient. The following figure displays the relevant information, the CSV file of battle data has.

When the mongoimport command is executed, the files are in bulk are imported into the database.
30

Figure 6.9. Mongoimport command for importing documents.

### 6.4 CLIENT SIDE CONTROLLERS

Client side controllers play a major role in this project. Many aspects like querying data from server, saving the data to database, filters, pagination and many more options are taken care by the controller. Scope Variable has the data which can be used in the controller as well as in the HTML page.

Initially, the app is registered and the required libraries are also imported. The code snippet to register modules is given below.

```javascript
var webapp = angular.module('webApp', ['ngResource', 'infinite-scroll', 'ui.bootstrap']);
```

Figure 6.10. Registering modules.

To run the application without errors in the HTML pages, a few angular directives must be added to the application like ng-app = webapp; ng-controller, model. Using ng-controller, we assign a controller, which can be used in the view. Ng-model is used to store the data from the HTML page and can also be used to do some specific operations with the data.

### 6.5 QUERYING AND SAVING DATA

The following code has two functions that execute at the client side. The first one queries the results and stores it into a scope variable. It also specifies where the data must be loaded. Resource object handles the REST API data. The query function here, emulates a GET http request and requests the server, the list of data. This query is executed automatically when the page is loaded and finally, the data is stored at the specified URL.

The second part of the code is executed when we want to insert the data into the database and at the same time update the page without refreshing. Save function invokes the POST method.
Filters are one of the important components in our application. Using a filter the data is sorted and filtered, giving only the results from the selection. A function is written for calculating the range and is filtered accordingly. Also, search value is stored in an ng-model and is used as one of attributes to be filtered. The following snippet results in filtering the data.

Below figure represents the range of years to be selected and also a search query, if we want to know information about a specific battle, kingdom or a ruler.

```javascript
var battlegrounda = $resource('/api/battlemongodata');
battlegrounda.query(function(battleresults) {
    $scope.battlegroundaStored = battleresults;
});

function createBattlegrounda()
    $scope.battlegroundaStored.push({name:$scope.battlegrounda});
    console.log($scope.battlegroundaStored); // $scope.meetups.push({name:$scope.meetupName});
battlegrounda.nameOfBattle = $scope.battlegrounda.nameOfBattle;
battlegrounda.description = $scope.battlegrounda.description;
battlegrounda.yearOfBattle = $scope.battlegrounda.yearOfBattle;

function createBattlegrounda(battleresult){
    console.log(battleresult);
    $scope.battlegroundaStored.push(battleresult);
    $scope.battlegrounda = ''; 
});

function createBattlegrounda()
    $scope.battlegroundaStored.push({name:$scope.battlegrounda});
    console.log($scope.battlegroundaStored); // $scope.meetups.push({name:$scope.meetupName});
battlegrounda.nameOfBattle = $scope.battlegrounda.nameOfBattle;
battlegrounda.description = $scope.battlegrounda.description;
battlegrounda.yearOfBattle = $scope.battlegrounda.yearOfBattle;

function createBattlegrounda(battleresult){
    console.log(battleresult);
    $scope.battlegroundaStored.push(battleresult);
    $scope.battlegrounda = ''; 
});

<tr ng-repeat = "item in battlegroundaStored | yearbetween: yearStart:yearEnd | filter: querysearch | orderBy:sortType:sortReverse"
    ng-style="{'background-color': $index == selectedIndex ? 'lightgray': '}'" ng-click="selectTitle(item, $index)"
>
The aim of the project is to generate a dynamic content in a page rather than redirecting to a static HTML page. This is achieved by taking the name of the event or ruler and sending it to the database as input and then retrieving the data. The input taken, when the specified URL is hit, returns the results which are sent back to be displayed.

The above data is used to differentiate the type of data the user wants and calls. The corresponding functions are called thereby getting a response with the relevant information. The Ruler Information is displayed at **localhost:3000/RULERNAME**, whereas the URL for battle event is **localhost:3000/api1/BATTLENAME**. The following figures illustrate the URLs for destination pages.

![Image of localhost:3000/api1/Battle%20of%20Gaugamela](image1)

**Figure 6.15.** URL locations of dynamic generated content.
CHAPTER 7

OVERVIEW OF THE PROJECT

An overview of how the application and website co-exist with each other is discussed in this chapter. The below figure represents the GIS application, in which, by using the hotlink tool and clicking on a point in the map, displays a window giving the information about the event and linking it to the Website.

Figure 7.1. GIS application with hotlink tool.

The below page is the home page of the MEAN website where the information about different tabs and their functionality is mentioned in brief.
In this tab, when we can select a range of years, the list gets populated accordingly and displays the rulers in that period. A search box is also present to search about a ruler without refreshing the page. The table can be sorted either by the name or by their reign. When selected, a row gets highlighted and their details are also displayed on the right side.

Figure 7.3. Quick details page.
In this page, all the list of rulers are displayed along with their dynasties. Pagination is included and a search box is provided. By clicking the link on a specific ruler, redirection to a specific page, containing detailed information about the ruler is displayed.

Figure 7.4. Smart table of all rulers with pagination.

When a “know more“ link is selected the website redirects to the below page where the details of the ruler and the description is shown at localhost:3000/RULER_NAME. This page is equipped with a different controller.

Figure 7.5. Detailed information about the ruler.
This tab contains rulers listed in a timeline fashion, giving more specifics about their rule period and simultaneously, various dynasties can also be selected without refreshing the pages. A search box is present along with a side bar containing a list of 10 dynasties.

Figure 7.6. Timeline of different dynasties.

A list similar to the rulers is displayed upon loading the page. By selecting the link “know more” about the event, it gets redirected to the specifics of a battle.

Figure 7.7. Smart table about the battles.
The battle information is displayed in this page. All the event data is pulled from the database and is displayed at localhost:3000/api1/BATTLE_NAME.

Figure 7.8. Detailed view of the battle.

Figure 7.9. Add to database page.
Finally to save information to the database, the user accesses this page and fills it with the appropriate details and finally selects “Add to Database” button.
CHAPTER 8

SUMMARY

This thesis presents information about various kingdoms in the Hellenistic civilization and their battles using a GIS application along with an interactive website. Additionally, the battles represented here are specific to the period of 323BC – 31BC (post Alexander-The Great’s Death).

The GIS application represents the boundaries of the kingdoms and the battle locations. Each battle event can be explored using the hotlink tool, where information about the battle is displayed in another window along with a link, to redirect the user to a website displaying additional details. Many predefined tools along with custom tools are integrated into the application for a smooth flow.

The website is designed in such a way that the user has the ability to vary the range of years, which eventually populate all the rulers of the selected range. A timeline style is used to showcase different dynasties in the website. The Information about various battles and kings are displayed in smart tables with pagination. Further detailed information about the rulers and battles are also projected dynamically according to the selection.

8.1 DIFFICULTIES

The Hellenistic kingdoms are spread over different regions and occupy vast boundary areas. Information about different kingdoms and their maps are not well presented in the Internet. The data collected for generating the shape files of the kingdoms needed extensive search from historical/educational websites. The Shapefiles are created using a Polyline Polygon Tool for GIS [19].

Also, the major part of the data which covers over 10 dynasties, lasting over 3 centuries are ruled by many great rulers which are displayed using the smart tables. There was a lack of information about these ruler since there were few websites showing such a
definitive data collection. The data is manually collected from the websites and is stored in a .csv file which is used as a source for inserting the same into the database. A lot of study was required due to the vastness of this topic. The process of creating the dynamic website using the latest technologies was another huge task to accomplish. A lot of time working on the website involved figuring out the interactive flow of its content and its dynamic display.

8.2 Future Scope

The application showcases information about the Hellenistic kingdoms using the MOJO Java and MEAN technologies. The application can be further implemented using future technologies as we progress further reaching out to a higher audience. A few enhancements that could be done are listed below.

- The application can be made lightweight using JavaFx libraries, a product of ESRI.
- This application can be developed as a smart phone app as well using Android and iOS SDKs.
- A deeper study could reveal more information about the kingdoms and events that occurred in the specific time which can be shown as a part of timeline.
- Integration of google maps API can show boundaries in a more interactive way.
- Family tree representation of different kingdoms can be incorporated into the application.
- The range of years can extended as well.
REFERENCES


