ISSN (e): **2250 – 3005** || *Volume, 07* || *Issue, 05*|| *May – 2017* ||
International Journal of Computational Engineering Research (IJCER)

API Design Fostering Payment Methods

[1]bandita Das, [2]ritu Priyunka,
Gandhi Institute of Excellent Technocrats, Bhubaneswar,India
Majhighariani Institute of Technology and Science, Rayagada, Odisha, India

ABSTRACT- The design of API patterns has grown exponentially over the past few years as compared to the old agemeaning that the development of APIs has become of essence than back in the days. This is a research projectthat will look in the field of the Application Programming Interfaces (API) and be able to apply the techniquesused, to the main functionality of the development of car hire management systems in Zimbabwe and abroad.The author is going to research mainly about REST APIs, the integration of payment methods so that at the endatheoreticalapplication can beusedtocomeup with aprototypeoftheAPI thatcan beusedin abovementioned field. Integrated in the API are various payment methods such as (Ecocash, Telecash) and ZIMRAtax module. Mainly about REST APIs and integration of payment methods with a view of coming up withcomprehensiveandeasy touseAPIs[3].

I. INTRODUCTION

TheresearchprojectismainlyfocusedonApplicationProgrammableInterfacesthatarebeingusedwiththemajoraimofc omingup with a defined API pattern fostering payment methods and revenue collection for car hire companies. Application ProgrammableInterface (API) is a set of protocols and routines as well as the tools that are used to build software applications.it specifies theinteraction between components in a software. APIs have become the most fascinating section of programming. The fact that theyaddress programmer challenges rather than customer requirements makes it both fascinating and novel. As the software tradeand the open-source effort continually grows, the number of public APIs is also gradually increasing. The good thing aboutAPIs is that they can improve the development speed thus their popularity. They also contribute a standardized and higherquality software and at the same time increasing the reusability of software. APIs are designed to be used by programmers andmost of the programmers are finding it easy to use APIs rather than developing their own program right from scratch. Hencethis paper is aimed at reviewing the critical points of the current knowledge about APIs. Generally, APIs have a number ofclassifications which are based on what purpose they serve and/or how they work. There are internal APIs which are usedstrictlywithintheorganizationor firm,thereareIsoexternalAPIs whicharespecificallydesigned externallyforcustomers.

The most important questionthat a developershould ask themselveswheneverthey planondesigning anAPI is what arethey trying to achieve with that API, this will help to stay focus on the design and not end up getting carried away because themainpurposeofdesigningAPIsistomaketheprogrammeras successfulas possibleintheleasttimegiven.
he orientation for APIs is to think about design choices coming from the application developer‟s opinion. The primarydesign principle when crafting your API should be to capitalize on developer‟s productivity and accomplishment, hence it iscalledthe“PragmaticRest.

II. RELATEDWORK

CarQuery API [13] is a JSON based API meant for retrieving a detailed car and truck information which include the year,make, the model, trim,and also its specifications. The CarQuery API is so simple to use suchthatitcanbe equated toincluding a JavaScript file, and as well as inserting a few lines of code in your page. There is room for the developer to writetheirownJavaScriptoraserver-sideAPIinteraction.

Fuel API [14]. This is an API that was developed around the principles of REST. Allows one to search, download and saveimages andvideosofvehicles, interiorandexteriorfeatures.

The Locu Data API [15] provides real-time access to Locu's dataset of semantically-annotated venue and price

list information. TheAPI enables developers with the ability to make location based queries for detailed venue, price list, and price list item information.The Locu Data is divided into two APIs that is the API for matching a third party venues to Locu‟s, meaning that partners arepermitted to send in their venues and in return will information that Locu knows about the venue. The search API supports thefetchingof venuesby multiple typesof IDs, the date added or changed, looking for menus. The Locu API haswhat they call thevenue object which is the core object for matching constructed venues by partners, {"fields": ["location", "menus"]}, this venues objectsearchesforthelocation detailsandalsomenuswhichare found inthatvenue. ItsupportstheCURLandJSON formats.

The Carvoyant API [16]. This is a RESTful web API which allows customers and the partners to access some Carvoyant dataprogrammatically. All the API calls must be made over the SSL and the accessto the system iscontrolled by an authenticationprocess OAuth2 implementation. To access the API you use the URI: https://api.carvoyant.com/v1/api. All the requests areexpected to include Authorization header that contains the bearer token that has been generated through the OAuth2 granttypes. With APIs everything is a resource hence the API is structured in terms of resources. The various requests that can bemade against a resource include PUT, DELETE, GET, and POST. It supports JSON format. It has the capabilities of giving aJSON successresponseformatdistinctfromaJSONerrorformat.

III. METHODOLOGY

The author started with a very simple design for the solution which allowed more extensions and additions to be applied duringstages of development. The design mainly dealt with views on how users will be interacting with the solution [8]. The authormainly used four controller patterns which allow him to split the solution into different parts so that it will be easier to changeparts of the solution without affecting other parts of it. Model refers to methods for accessing and modifying states whereasViews are those that renders contents of model for user and Controllers translate user actions (interactions with the view) intooperations on the model for example button clicks and menu selection. The author saw it important to use Model-View-Controllerdesignpatternas it allowedhim to separate majorcomponents thatcanbe workedoninrelative isolation.Therefore, the author mainly started with the design of different views on how he would like users to interact with the solution[1].

Later other additions were made to the design to add more functionality and ease of use with regards to the API and duringdesigningofthesolutionviews,theauthordecidedtonamethesolutionAPIasMorthparkAPI.

The system architecture is the conceptual model that defines the structure, behaviour and more views of a system. Anarchitecture descriptionis a formal descriptionand representationof a system. The system was represented following theModelViewcontrollerdesign pattern.[5]Thepattern‟stitleisacollation ofitsthreecoreparts:Model,View,andController.

Avisualrepresentationofacompleteand correctMVC patternlookslikethefollowingdiagram:

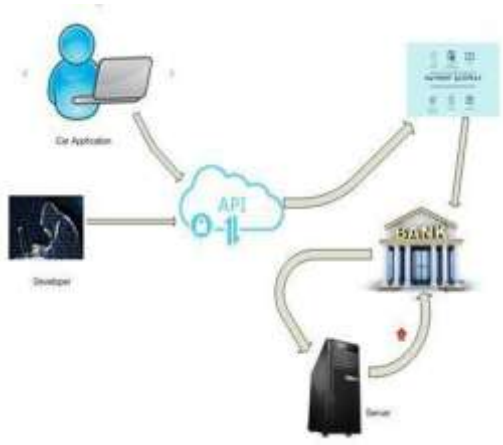

Fig.1.SystemArchitecturalDesign

The diagram shows the flow layout of data, how it is passed within each component and how the relationship within eachcomponentworks.

A. MODEL

The Model is the name given to the permanent storage of the data used in the overall design. It must allow

access for the datato be viewed, or collected and written to, and is the bridge between the View component and the Controller component in theoverallpattern.
One important aspect of the Model is that it‟s technically "blind" – by this we mean the model has no connection orknowledge of what happens to the data when it is passed to the View or Controller components. It neither calls nor seeks aresponsefromtheotherparts;itssolepurposeistoprocessdataintoitspermanentstorageorseekandpreparedatatobepassedalongtotheotherparts. [2]

The Model, however, cannot simply be summed up as a database, or a gateway to another system which handles the dataprocess. The Model must act as a gatekeeper to the data itself, asking no questions but accepting all requests which comes itsway. Often the most complex part of the MVC system, the Model component is also the pinnacle of the whole system sincewithoutitthereisn‟taconnectionbetweentheControllerandtheView.

B. VIEW
The View is where data, requested from the Model, is viewed and its final output is determined. Traditionally in web appsbuilt using MVC, the View is the part of the system where the HTML is generated and displayed. [3] The View also ignitesreactions from the user, who then goes on to interact with the Controller. The basic example of this is a button generated by aView,whichauserclicksandtriggersanactionintheController.

C. CONTROLLER
The final component of the triad is the Controller. Its job is to handle data that the user inputs or submits, and update theModel accordingly. The Controller‟s life blood is the user; without user interactions, the Controller has no purpose. It is theonlypartofthepatterntheusershouldbeinteractingwith.

The Controller can be summed up simply as a collector of information, which then passes it on to the Model to be organizedforstorage,anddoesnotcontainanylogicotherthanthatneededtocollecttheinput.TheControlleris alsoonlyconnectedtoa single View and to a single Model, making it a one way data flow system, with handshakes and signoffs at each point of dataexchange. [5]

It‟s important to remember the Controller is only given tasks to perform when the user interacts with the View first, and thateach Controller function is a trigger, set off by the user‟s interaction with the View. The most common mistake made bydevelopers is confusing the Controller for a gateway, and ultimately assigning it functions and responsibilities that the Viewshouldhave(thisisnormallyaresultofthesamedeveloperconfusingtheViewcomponentsimplyasatemplate).Additionally,it‟sacommonmistaketogivetheControllerfunctionsthatgiveitthesoleresponsibility ofcrunching,passing, and processing data from the Model to the View, whereas in the MVC pattern this relationship should be kept between theModelandtheView.[9].

D. USECASEDIAGRAM
Below is a use case diagram depicting use of the developed system. The system has two users namely the administrator andthecostengineer.Figure4basicallyshows howthesetwousers willbemanipulatingthis systemsvarious functionality.

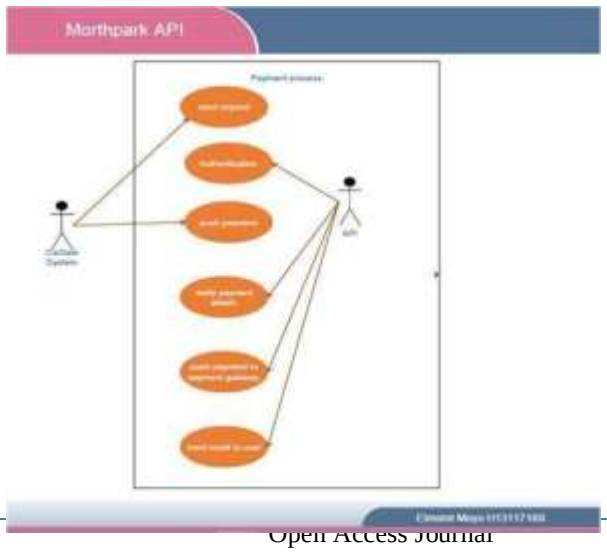

Fig.2.Usecasediagramfor the system

E. IMPLEMENTATION AND TOOLS The development tools that were used for the development of MorthparkAPIconsistedofthefollowing

1) JAVASCRIPT

JavaScript is the client-side scripting language used during the development phase. In certain circumstances, JavaScript had to beusedinsteadofPHP.Onesuchinstanceincludesdisplayingpopupwindowstoalertusersoferrorsindataenteredduring validationof input or possible loss of data when a delete operation is executed [8]. The fact that the language works in a run-time environmentwhich is especially true in web browsers, makes JavaScript suitable for offering the required services that a server-side languagecannot. However, JavaScript is not used to pass sensitive data such as passwords in view that the codes are executed at the client sidewherethesystemcan bevulnerabletomaliciousattacks[1].

2) HTML

HTML, acronym forHypertext Markup Language, is a markup language used to describe the formatting of textin adocument. It is useful in the sense that it allows text to be structured according to its purpose, namely as a heading, paragraphand so on. This is accomplished by writing the HTML in „tags" that describes to the web browser how the text is to bedisplayed. A scripting language such as PHP and JavaScript can be easily embedded in HTML to enhance the functionality ofHTML[11].

3) SMWEBCREATE

SMWEBCREATE is short for Sadomba-MahariWebcreate which is a platform for helping in the design and development ofwebsites. The functionality of SMWECREATE is similar to that of ADOBE DREAMWEAVER where graphics and webcomponents can be dragged and dropped to the appropriate location at the same time it automatically generate the code whereadditionalcodemanipulationcanbedonebytheprogrammer[10].

4) DEVELOPMENTPLATFORM

The windows 8 operating system was selected as platform for developing MORTHPARK API. The fact that the potential usersarealreadyusingandarefamiliarwiththeWindows 8environmentplayedanimportantroleintheselection[8].

5) DATABASE

MySQL is a relational database management system (also known as an SQL Database Server) which is widely used around the globeduetoitbeing open-sourced. Most SQLserversprovidereliabilitybutnot easeofuseunlikeMySQL
[6].MySQL is also mostly platform independent which means it can run on most operating systems such as Windows andLinux[6].

FUTURE WORK

The system can be further developed to cover a wide domain range not only focus on car sale systems. The system can befurtherdevelopedtogenerateAPIcodes tobeintegratedwiththecarsalesystemas anotherpaymentmethod.

CONCLUSION

Inanutshell,thesystemdesignandimplementationhasbeenasuccessasalltheobjectiveshavebeenmetdespitethesystem developerencountering somechallenges in theintegrationofvariouspaymentsmethods in thesystem.

REFERENCES

[1]. Li,L.,Chou,W.,Zhou,W.,&Luo,M."Designpatternsandextensibilityofrestapifornetworkingapplications". IEEETransactionsonNetwork andServiceManagement,2016.
[2]. EllisB.,Stylos,J.,&Myers,B."ThefactorypatterninAPIdesign:Ausabilityevaluation".Proceedings-InternationalConferenceonSoftwareEngineering, 2007.
[3]. Mulloy,B."WebAPIDesign– CraftingInterfacesthatDevelopersLove",36.2012
[4]. Stylos,J.,Myers,B.,Stylos,J.,&Myers,B."MappingtheSpaceofAPIDesignDecisions".2007
[5]. Woods,D.,Thurai,A.,Dournaee,B.,&Musser,J."Enterprise-classAPIPatternsforCloud&Mobile",(May),15. 2012
[6]. Zhou, W., Li, L., Luo, M., & Chou, W. "REST API design patterns for SDN northbound API". Proceedings -

2014IEEE28thInternationalConference onAdvancedinformationNetworkingandApplicationsWorkshops,IEEEWAINA2014.
[7].A.Meiappane,J.Prabavadhi&V.PrasannaVenkatesan."Strategy
[8]. Pattern :PaymentPattern". (2012).
[9]. Tak,P.,&Tang,P.DFTI"ANewInterfaceforFastFourierTransformLibraries",31(4),475–507.(2005).
[10]. Santhiar,A.,Pandita,O.,&Kanade,"A.MiningUnitTestsforDiscoveryandMigrationofMathAPIs",24(1).2014
[11]. Reed,J.A.,&John,N."Improving theAircraftDesignProcessUsingWeb-BasedModelingandSimulation",10(1),58–83. (2000)
[12]. RoyThomasFielding, "ArchitecturalStylesandtheDesignofNetwork-basedSoftwareArchitectures". 2000
[13].http://carvoyant-api.readthedocs.io/en/latest/getting-started**[14].**www.carqueryapi.com/
[15].fuelapi.com/
[16].https://dev.locu.com/
[17].https://carvoyant-api.readthedocs.io/