

Florida International University
School of Computing and Information Sciences

CIS 4911 - Senior Capstone Project
Software Engineering Focus

Final Deliverable

VoterCloud
Team # X

Team Members

Eldar Feldbeine.
Raul Garay.

Product Owner: Gus Monge.

Instructor: Masoud Sadjadi.

Copyright and trademark notices, restrictions on copying or distributing the documentation, information for contacting the issuing organization (reader's comments), warranties, contractual obligations or disclaimers, and general warnings and cautions.

Abstract

Communities flourish when their citizens are consistently engaged in their governments. In particular, local governments can have an even bigger on the day-to-day lives of the citizens they represent than larger scale ones. However, local governments are notorious for having a dearth of community involvement. Citizens can be apathetic, uninformed, or misinformed leading to a government that is not representative of its citizens. VoterCloud is a mobile application that focuses on bridging this gap between citizens and their governments. It intends to address the issue of apathy by connecting its users and politicians in a social environment as well as the issues of being uninformed by providing election and politician information.

Table of Contents

INTRODUCTION	5
CURRENT SYSTEM	5
PURPOSE OF NEW SYSTEM	6
USER STORIES	7
IMPLEMENTED USER STORIES	7
PENDING USER STORIES	32
PROJECT PLAN	33
HARDWARE AND SOFTWARE RESOURCES	33
SPRINTS PLAN	35
SYSTEM DESIGN	48
ARCHITECTURAL PATTERNS	51
SYSTEM AND SUBSYSTEM DECOMPOSITION	52
DEPLOYMENT DIAGRAM	53
DESIGN PATTERNS	55
SYSTEM VALIDATION	56
GLOSSARY	77
APPENDIX	78
APPENDIX A - UML DIAGRAMS	78
<i>Static UML Diagrams</i>	78
<i>Dynamic UML Diagrams</i>	80
APPENDIX B - USER INTERFACE DESIGN	97
APPENDIX C - SPRINT REVIEW REPORTS	105
APPENDIX D - SPRINT RETROSPECTIVE REPORTS	107
REFERENCES	

INTRODUCTION

Local government is at many times an afterthought in the public's mind. This is an issue because local government is the direct representation of a community. So many citizens do not realize that the decisions made by local government officials can often have a greater impact on the community than the decisions made in Washington D.C. Citizens are usually ignorant in the decisions made by local government and thus apathetic to their efforts. The following is a detailed description of the current state of the local government in relation to civic engagement. Then is a description of an application designed to assist citizens to get involved with their public officials and the decisions that affect their community.

Current System

Citizens who want to learn about their local politicians, their politicians' voting history, the laws proposed, and other local details that affect them currently get their information from newspapers or internet searches. Internet searches are usually done separately for each of these details. If it is election season, flyers, signs, and commercials are usually posted and distributed out around the respective election zones. These flyers and signs simply state to vote for or against a particular candidate or amendment without much or any description on the subject matter. Commercials are a form of communication that can provide more information than a flyer or sign could, but a candidate or an interest group pays for them. Thus, they tend to be extremely biased in the details revealed resulting in a skewed or incomplete message.

Once politicians are in office, there is very little civic engagement. Public meetings and hearings are held often to discuss amendments and issues that affect the politicians' zone. These meetings and hearings tend to produce a low public turnout. If a citizen wants to get involved and have their voice heard, they would need to either attend these meetings, write, email, or call their politicians. For most citizens, local politicians are usually not heard from again until election season.

Politicians also have a need to hear and understand the communities that they represent. The aforementioned meetings and hearings are open to the public so that the politicians can gauge public opinion and needs. In a technologically advancing world, politicians have taken to social media outlets such as Facebook and Twitter to interact with their community and voters. However, the assumption is that those who follow them are part of the affected community.

Many citizens also don't know who their local politicians are, what their roles are, and how their decisions affect the community. So it is difficult for politicians to properly reach out even when a citizen wants to get engaged.

Purpose of New System

The new system shall be an application that provides citizens information about all of the politicians that represent them, the amendments and laws being reviewed that affect them, and the political meetings and events taking place that they could attend. It should also provide a platform for politicians to reach out to the community they represent through surveys, videos, messages, and discussion/communication forums.

Such an application would personalize the experience for the user and remove clutter that does not affect them. In order to personalize the experience for the user, the application would require the user's home zip code so that all publicly available information about their local government can be provided to them. With this information, the user can be up to date on anything that affects them. To promote consistent engagement, the application should also have the ability to notify the user (in whatever format the user prefers) of any reminders for deadlines, meetings, events, etc.

The system must also allow politicians to reach out to the users of the application so that engagement becomes a two-way street and citizens. Politicians could send out surveys in order to gauge the community opinions. It should also let politicians schedule open forums that allow users to interact and converse with the politicians. This would allow those who do not have time to attend the publicly held meetings to still get involved.

Finally, the goal is to provide focus and knowledge to citizens' local civic opportunities and to reduce the apathy that exists towards local government. Citizens would feel empowered and politicians would be more knowledgeable about the people they represent.

USER STORIES

This chapter describes the main details and planning involved in creating the VoterCloud project. It composed of six sprints , in which each sprint describes the work taken in two weeks. As you dive into each sprint and each user story you can discover the functionalities implemented throughout this semester and the main logic description (using UMLs and other diagrams). The user stories chapter is perhaps the most important chapter of all , since it's describe our work in terms of logic and timing. Each user story is described in terms of description, acceptance criteria and diagrams to explain the concept and the logic. The team members expressed their talent and skills in implementing the use cases(user stories) as well as in describing them each sprint review. The order of the following user stories are in the order of their implementation, in which the first user story is the one implemented first for the first sprint and so on with the rest of the user stories and sprints. For complete picture of the system logic and planning please also look on the UML sections as it directly related to this section in describing the implementation and work involved.

Implemented User Stories

User Story # 703 - Setup Meteor Hosting

Description:

- As a VoterCloud team member, I want to deploy to a server and test the application in a simulated in environment so that I can complete production ready features.

Acceptance Criteria:

1. Ability to deploy new code to meteor.com host
2. Client code can communicate with server code in 2-way communication.

User Story #677 - Menu

Description:

- As a voter, I need a navigational menu, so that I can navigate through the application easily.

Acceptance Criteria:

1. Menu can be reached from any page.
2. Menu has list of all pages in the app.
3. Menu can appear and disappear on command (unobtrusive).

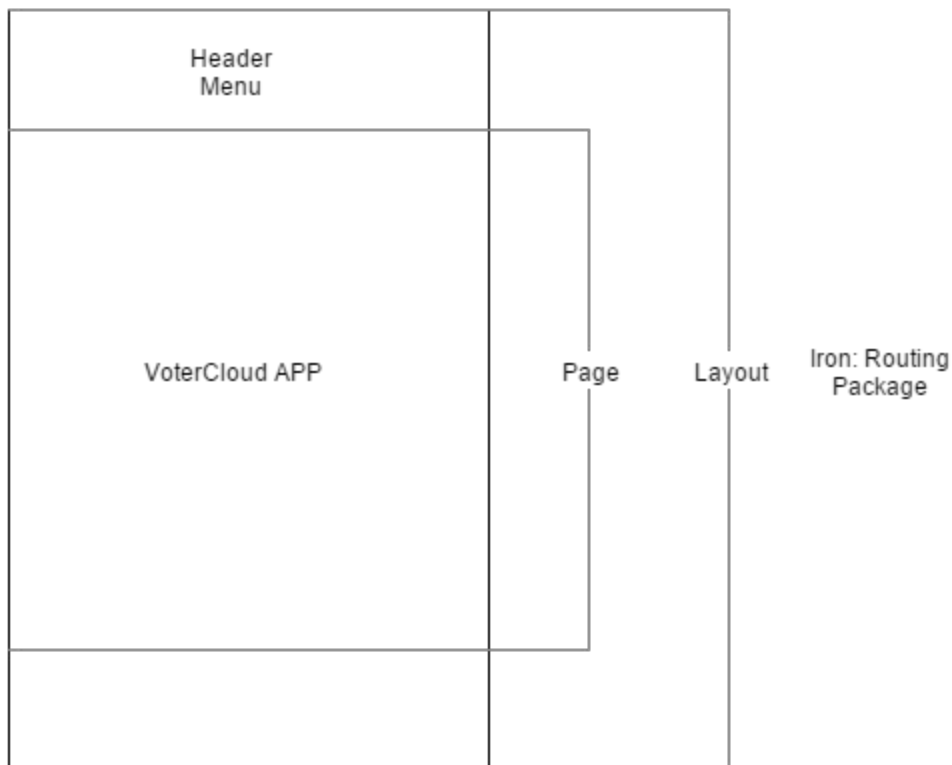


Figure 1.00 - Layout of the app as described in Iron: Routing Package.

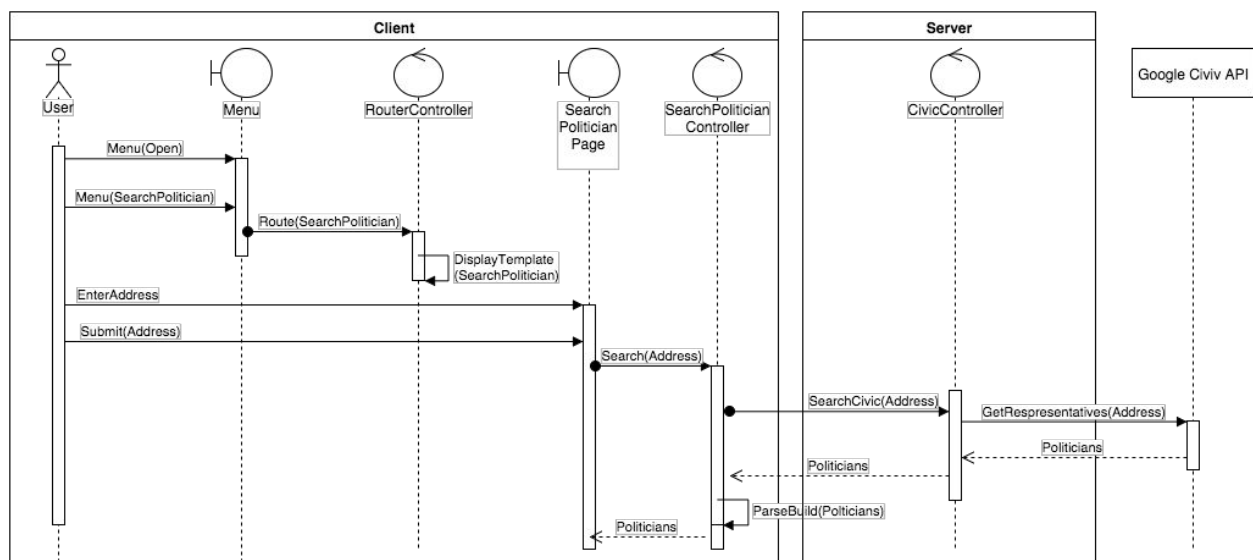
User Story # 673 - Search politician

Description:

- As a voter, I would like to search politician, so that i can find and explore more information about politicians.

Acceptance Criteria:

1. Ability to enter address and submit it.
2. see the corresponded politician profile and information.

**Figure 1.01** - Sequence diagram.

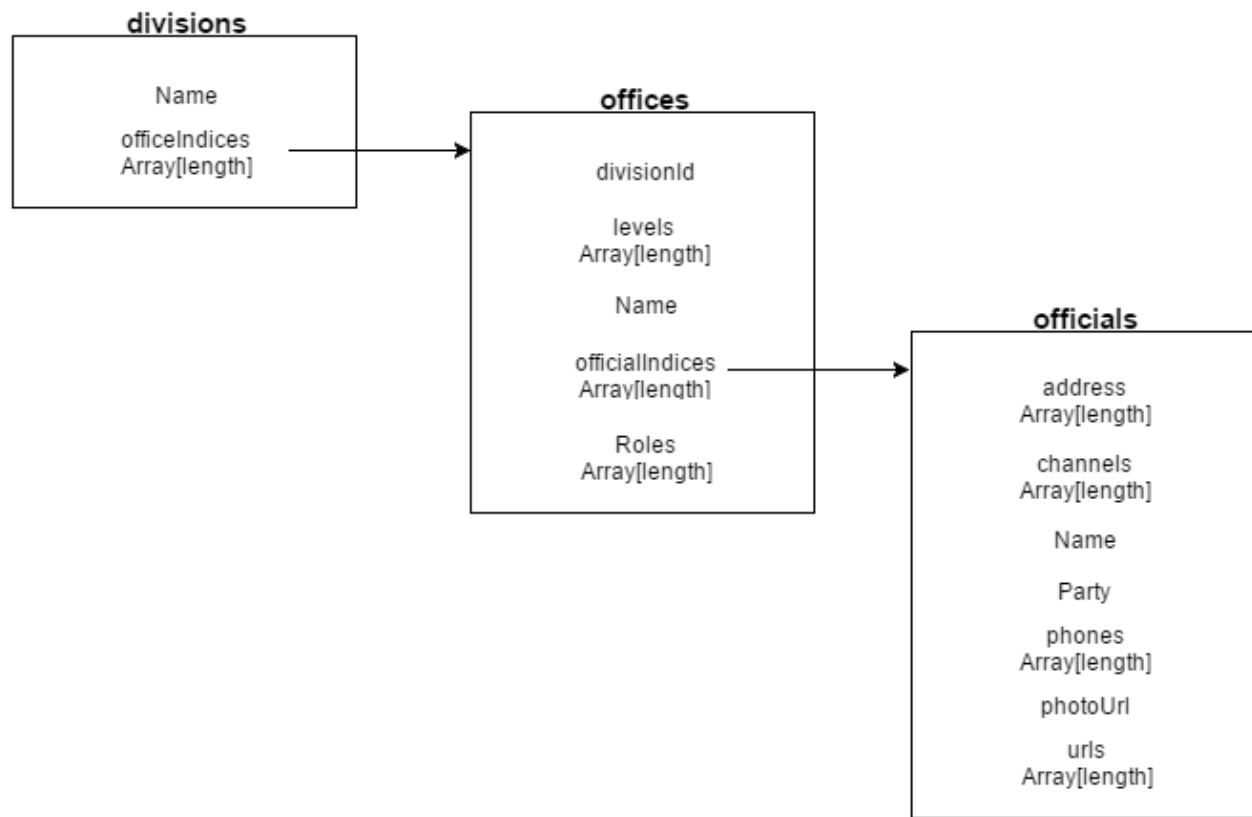


Figure 1.02 - Json file result of google civic api.

User Story # 713 - Elections

Description:

- As a voter enthusiast, I would like to know the upcoming elections, so i could make better decisions and get involve more.

Acceptance Criteria:

1. See upcoming elections based on areas.
2. be able to redirect to registration page for more information about the election.
3. be able to register and pull information about specific election.

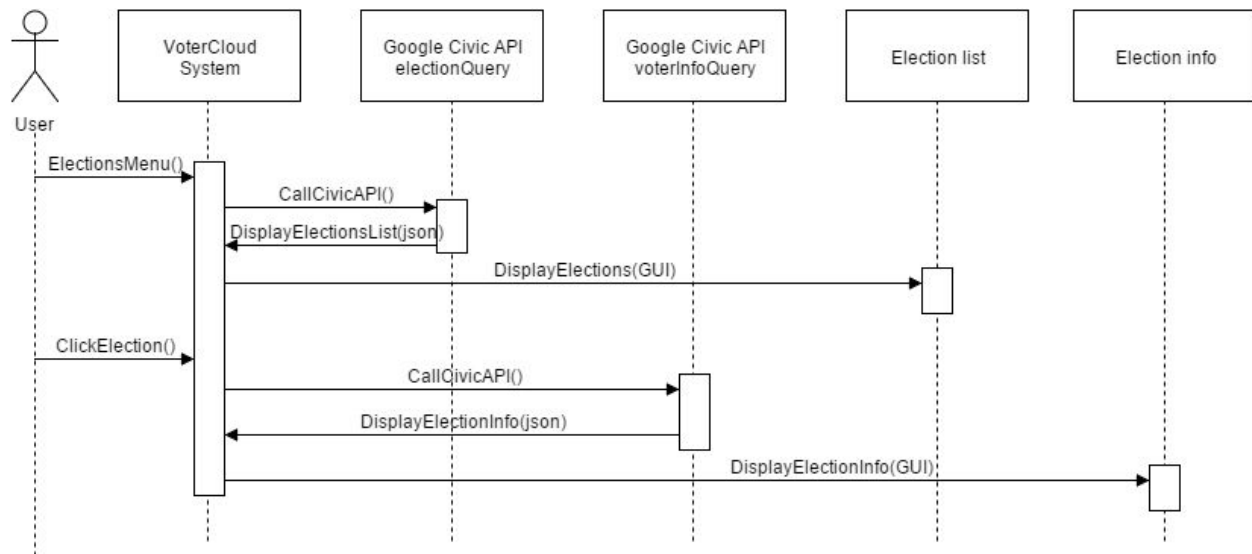


Figure 1.03 - Sequence diagram of the election user story.

User Story # 710 - Search by location**Description:**

- As a user, I would like to search any service by my current address using GPS. So that i would save time writing my address and get precises information.

Acceptance Criteria:

1. get correct latitude and longitude.
2. convert coordinates to real address.
3. real time fast location search



Figure 1.04 - Diagram presents the logic of the search by location.

User Story # 715 - Representative image profile

Description:

- As a user, i would like to see the representative image profile, so that i would get known the person appearance and description.

Acceptance Criteria:

1. be able to see representative image profile from his facebook, twitter, wikipedia profile..
2. the image should be 150 px to 150 px match to the other images styles.
3. load time for calling the Facebook graph, Twitter, Wikipedia APIs should be minimize.

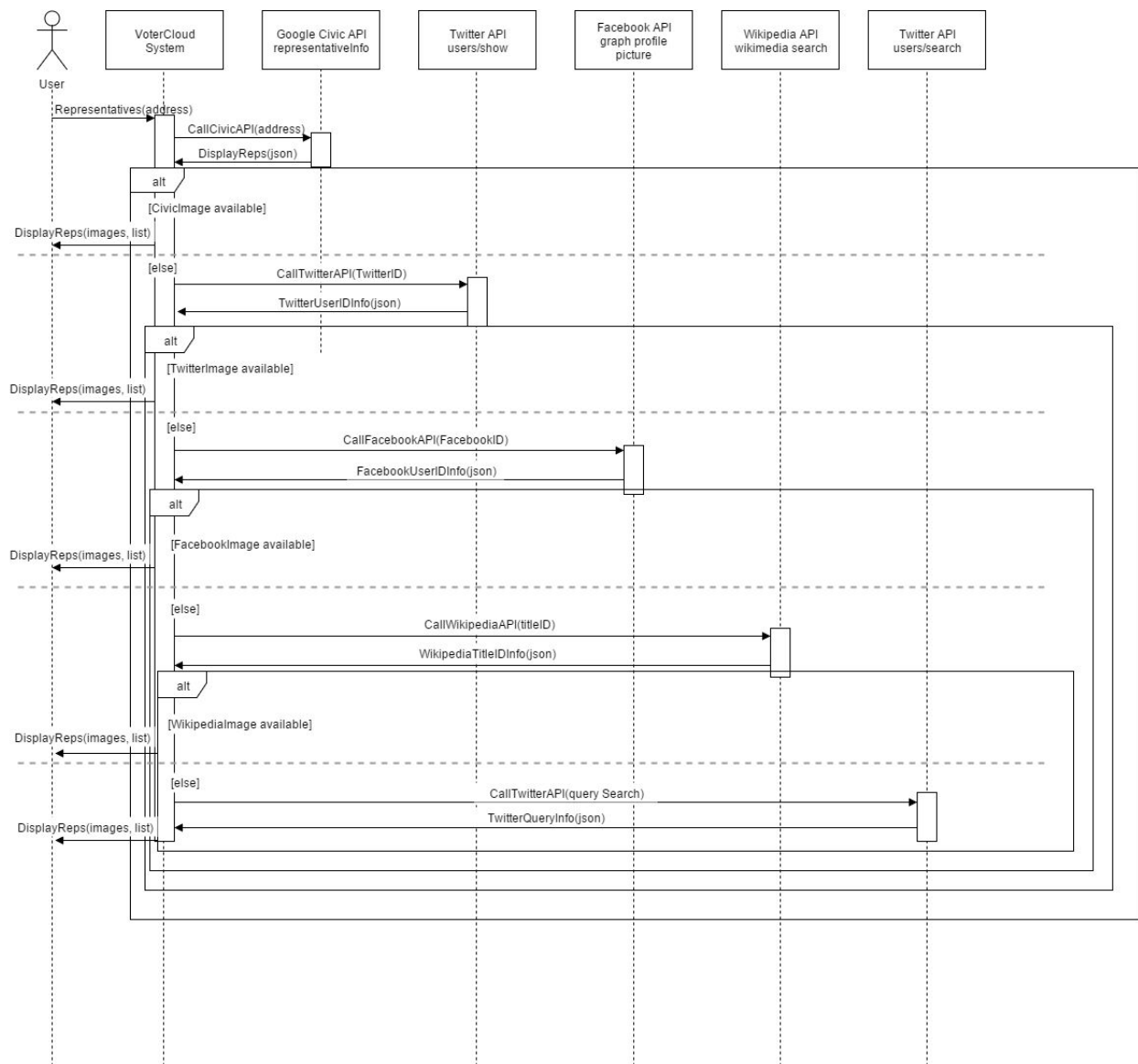


Figure 1.05 - Sequence Diagram shows the sequence of interaction and actions for repr image..

User Story # 675 - Surveys**Description:**

- As a voter, I want to fill a survey about my opinion, so that I can participate in polls.

Acceptance Criteria:

4. Forms can be filled.
5. The forms can be submitted.
6. The forms is added and collected as part of the statistical poll.

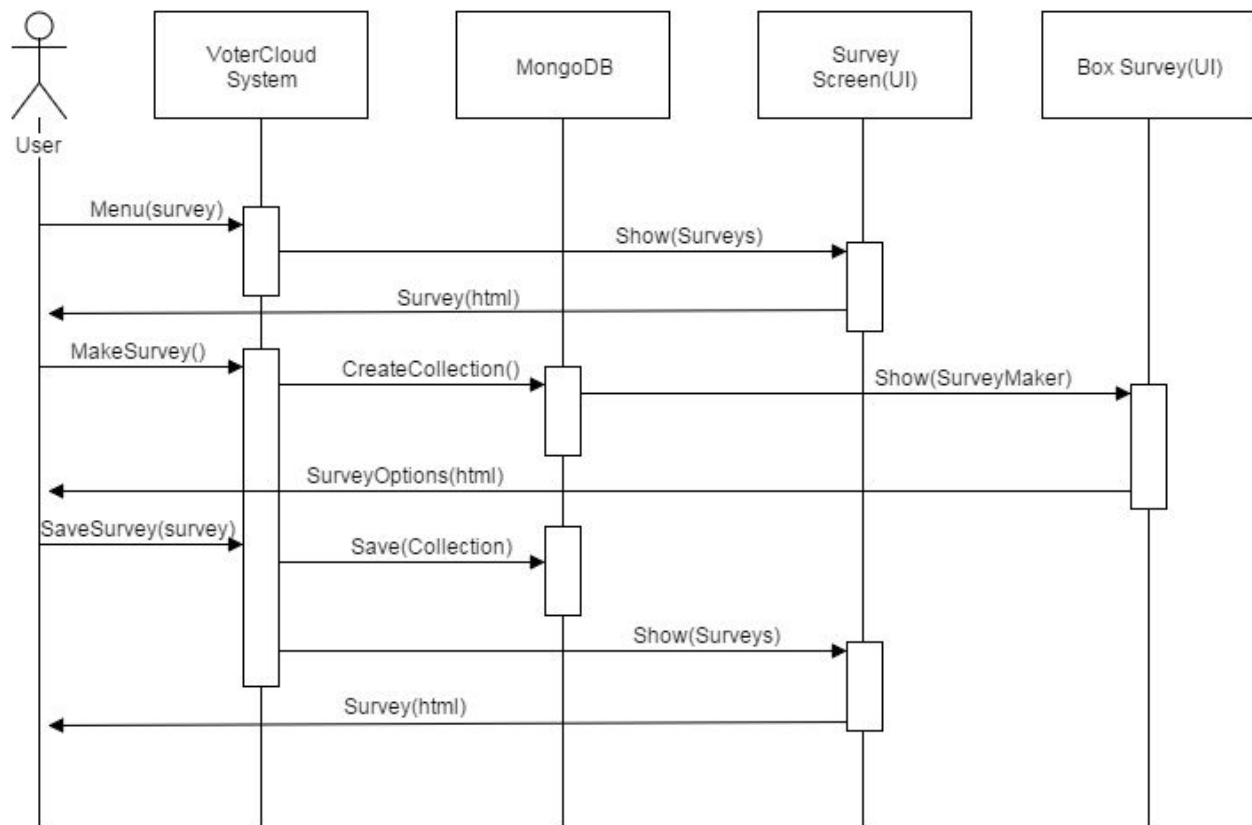


Figure 1.06 - The sequence diagram of the survey user story.

User Story # 722 - Petition**Description:**

- As a user, i would like to create petitions , so that other user can participate on petition actions/opinions.

Acceptance Criteria:

- Allow any registered user to create a petition.
- petition is dynamically updated.
- Details of each user that signed.

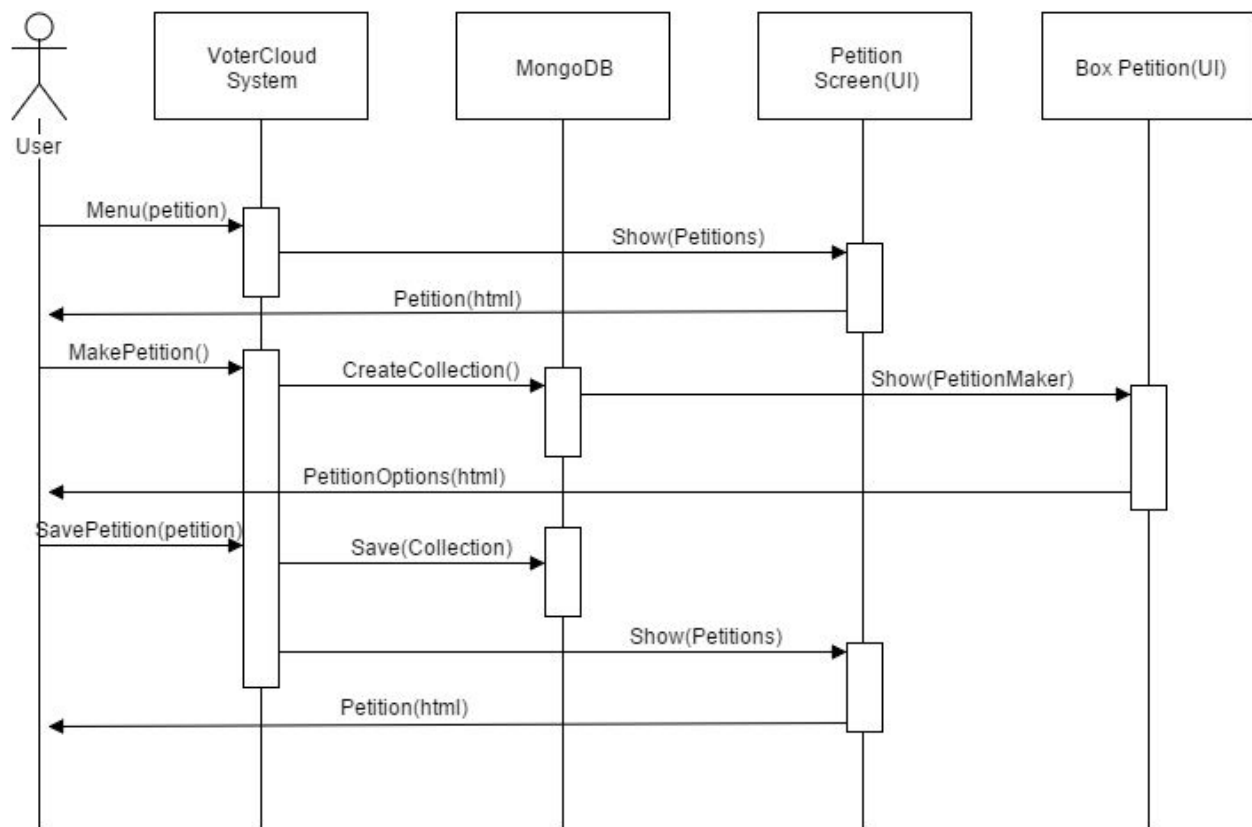


Figure 1.07 - The sequence diagram of the petition user story.

User Story # 724 - Chat Room

Description:

- As a voter, i would like to chat with other active voters, so that i can share my ideas and opinions.

Acceptance Criteria:

1. Reactive chat.
2. Live feedback.
3. Be able to see the user and date of each message.

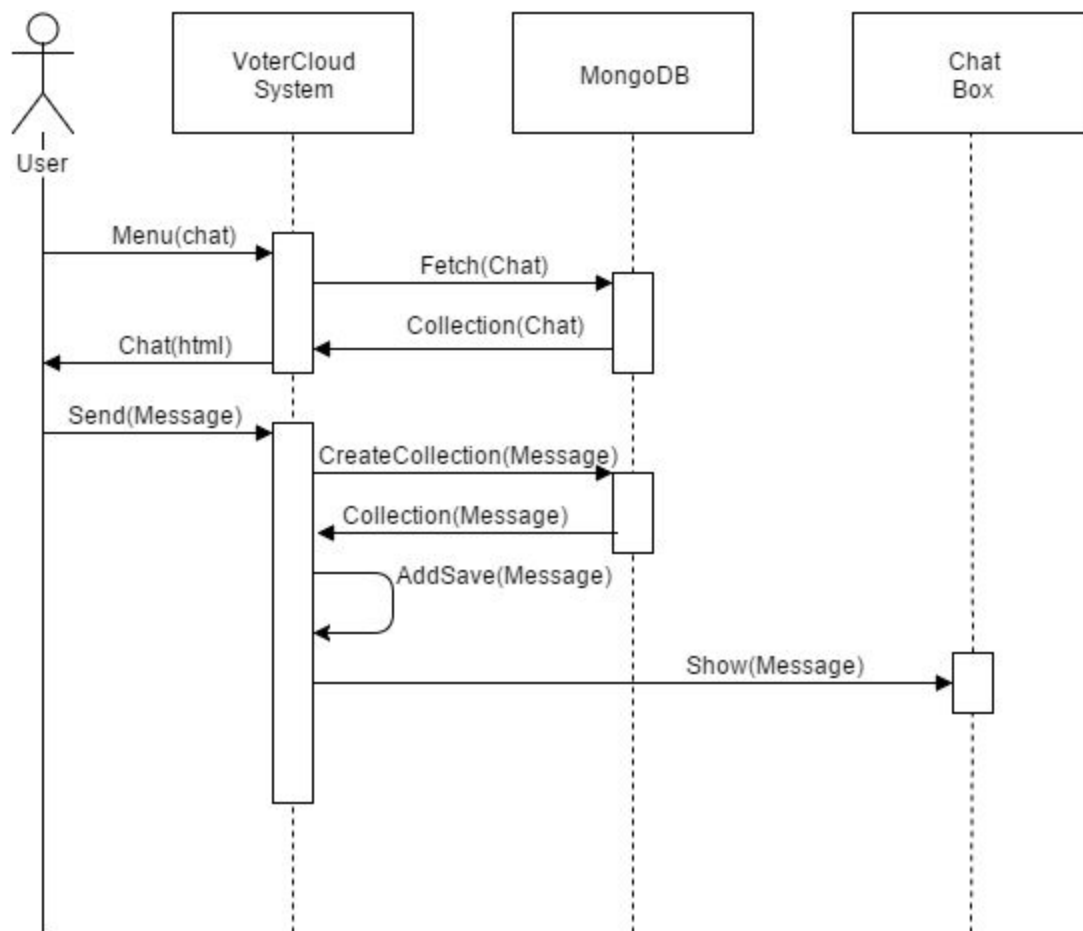


Figure 1.08 - The sequence diagram of the chat user story.

User Story # 725 - PDF generator and serving

Description:

- As a user, i would like to see the total results of the petition participants, so that i can find who vote for whom and be more engage.

Acceptance Criteria:

1. render the adobe reader file.
2. the content of the PDF should be the total supportive of the petition.
3. should be presented only when the limit of supportive reached.

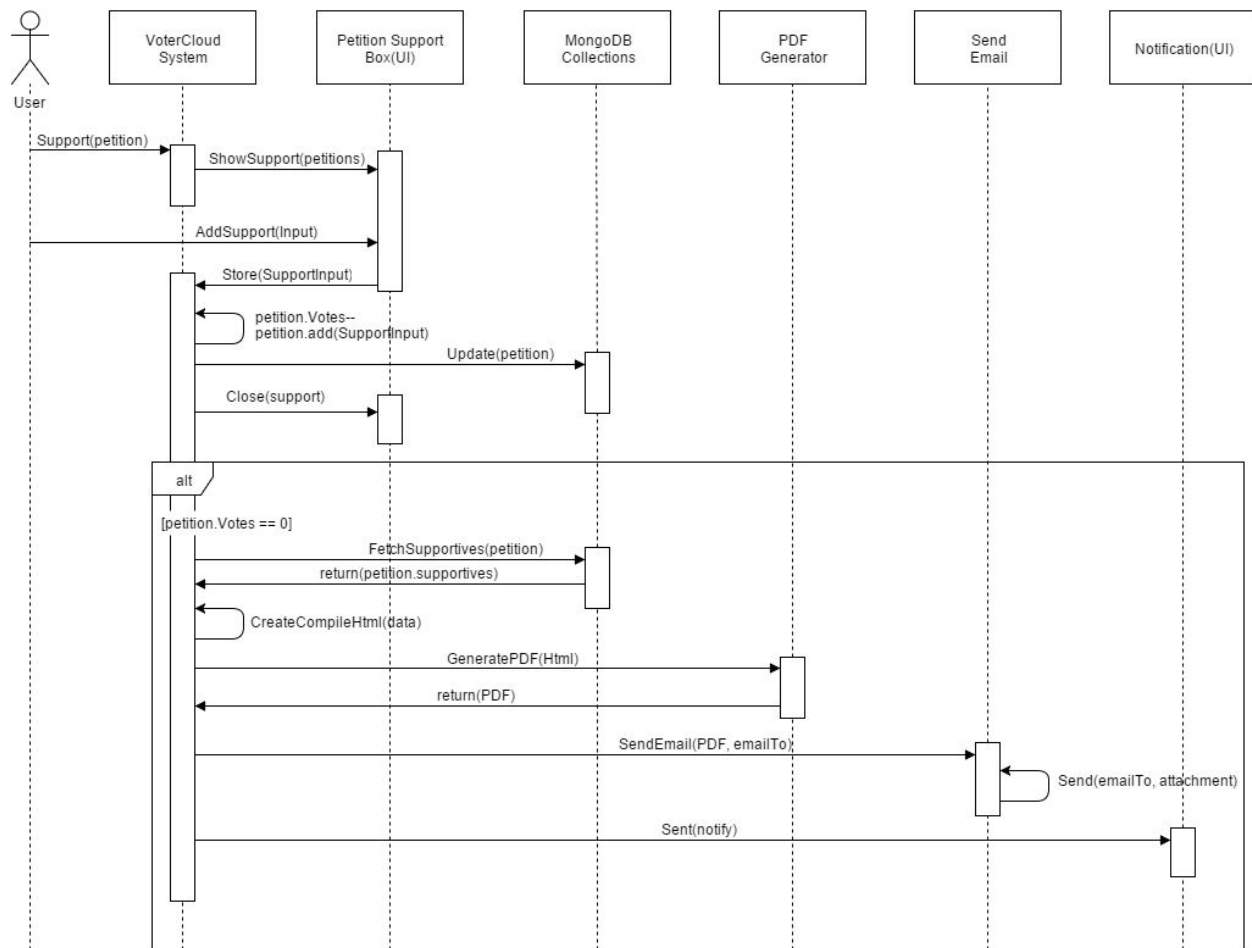


Figure 1.09 - The sequence diagram of the pdf generator and serving after petition support.

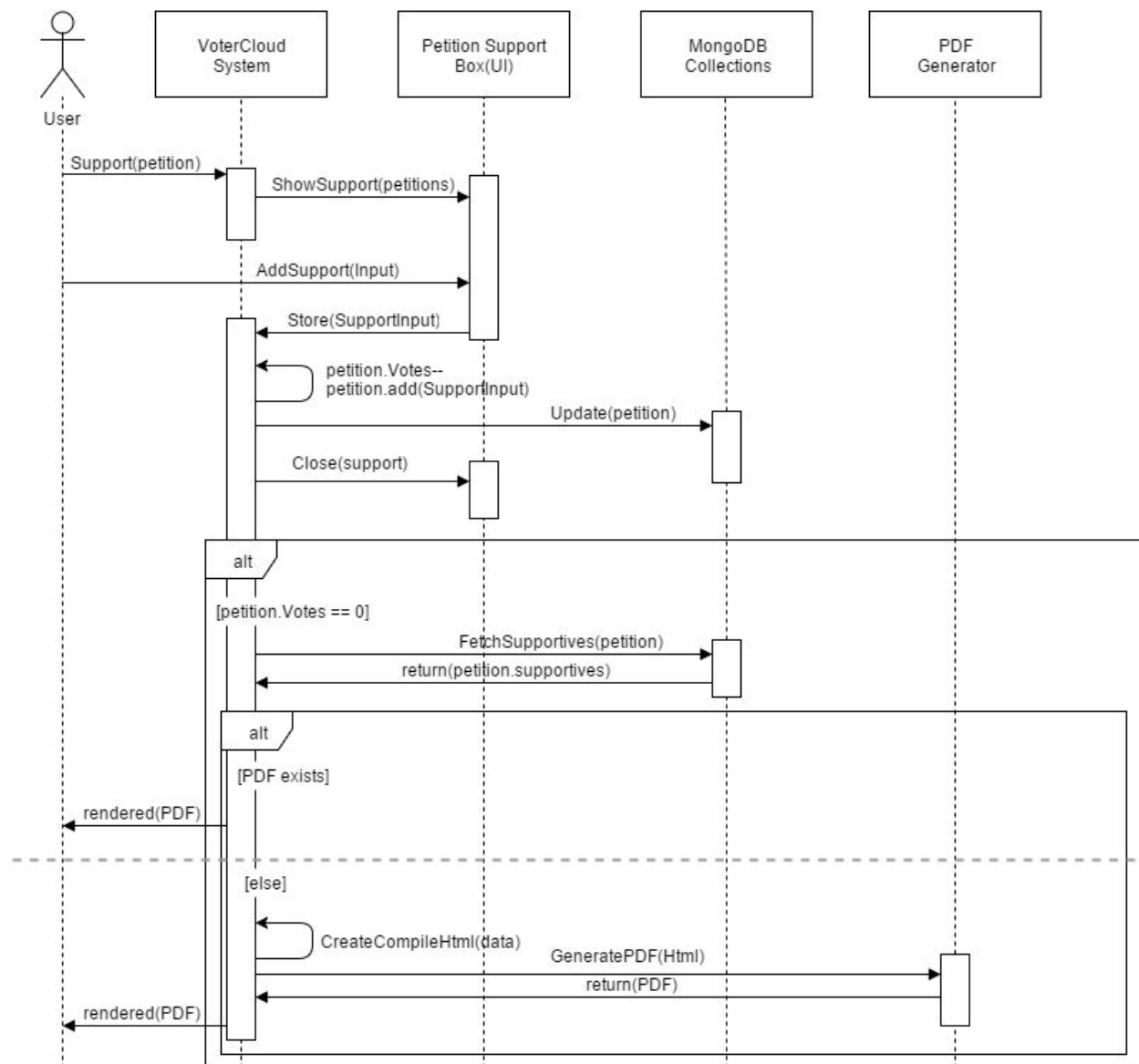


Figure 1.10 - The sequence diagram of the pdf generator and serving after petition support.

User Story # 723 - Signature

Description:

- As a voter and petition participate, I would like to sign my name on petition and other forms, so that my voice will be case appropriately.

Acceptance Criteria:

1. Be able to draw smoothly and signature.
2. Save the signature in compress and readable form.
3. Be able to store the image in MongoDB.

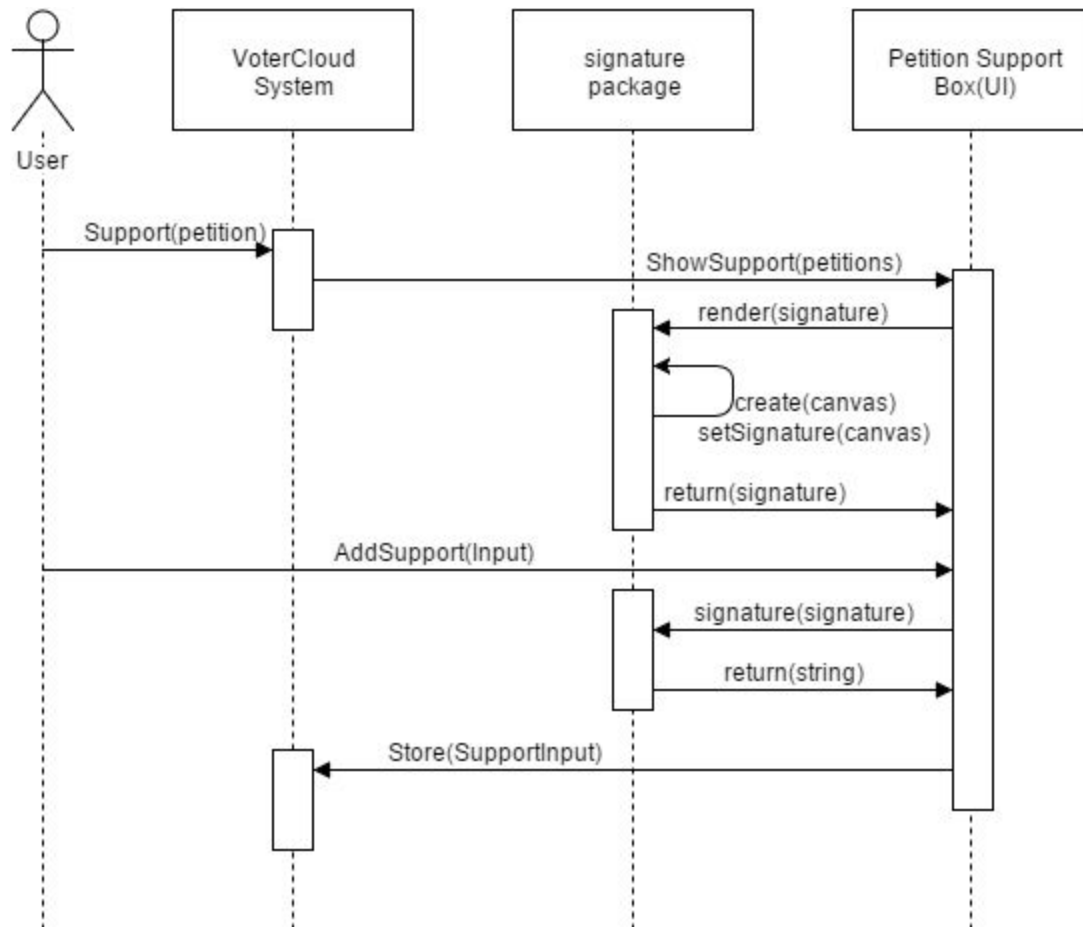


Figure 1.11 - The sequence diagram of the signature user story.

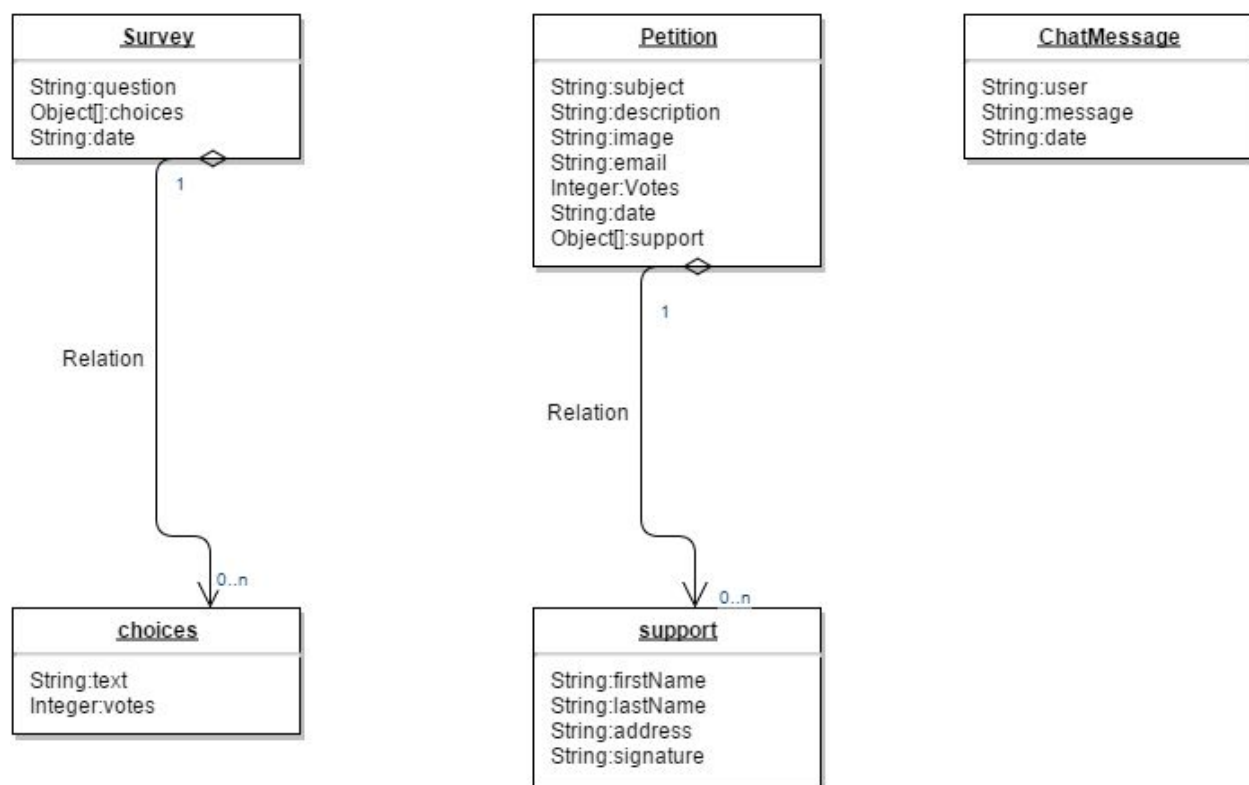
User Story # 724 - Create Hard Schema

Description:

- As an app user, I want security and consistency in my application when I trust the application to store my information.

Acceptance Criteria:

- Every data variable and command is accounted for in the schema.
- Writing to and reading from the database works the same.
- Validation controls are in place.



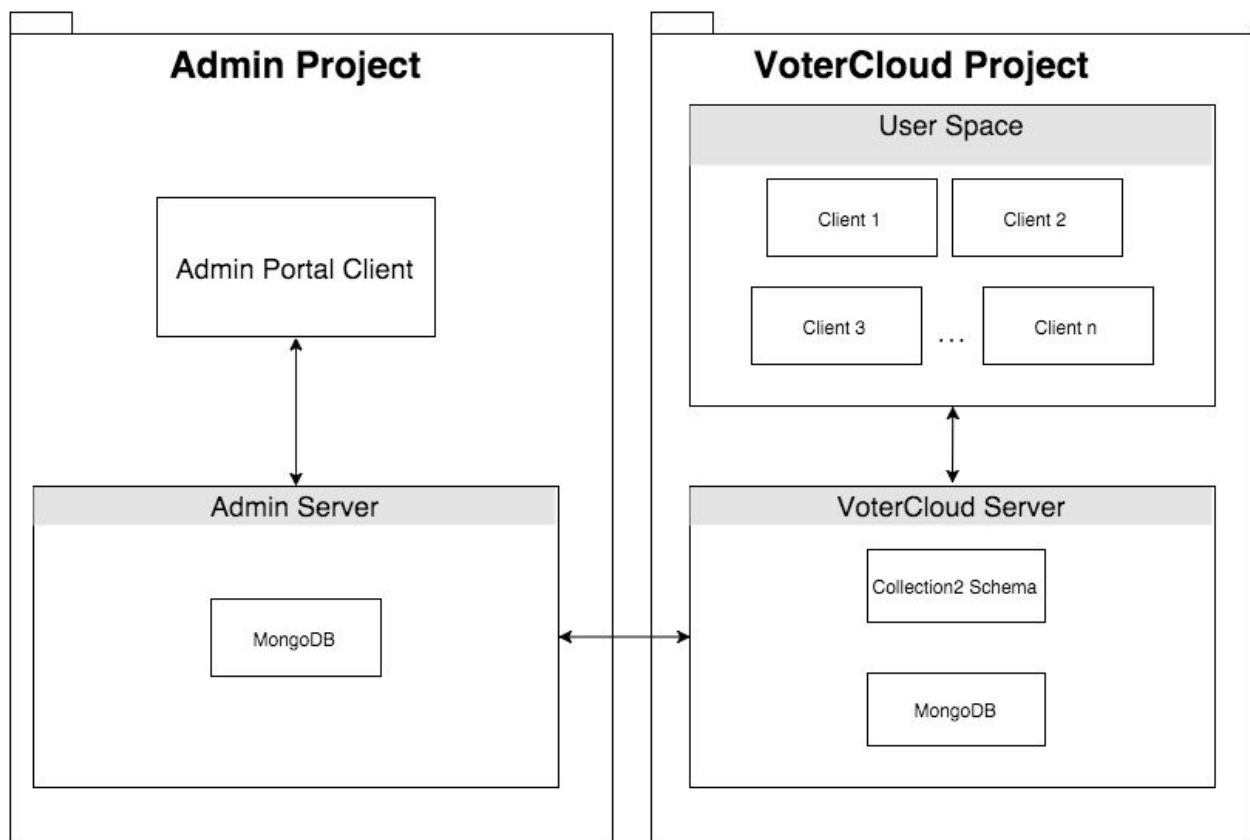
User Story # 725 - Admin Portal

Description:

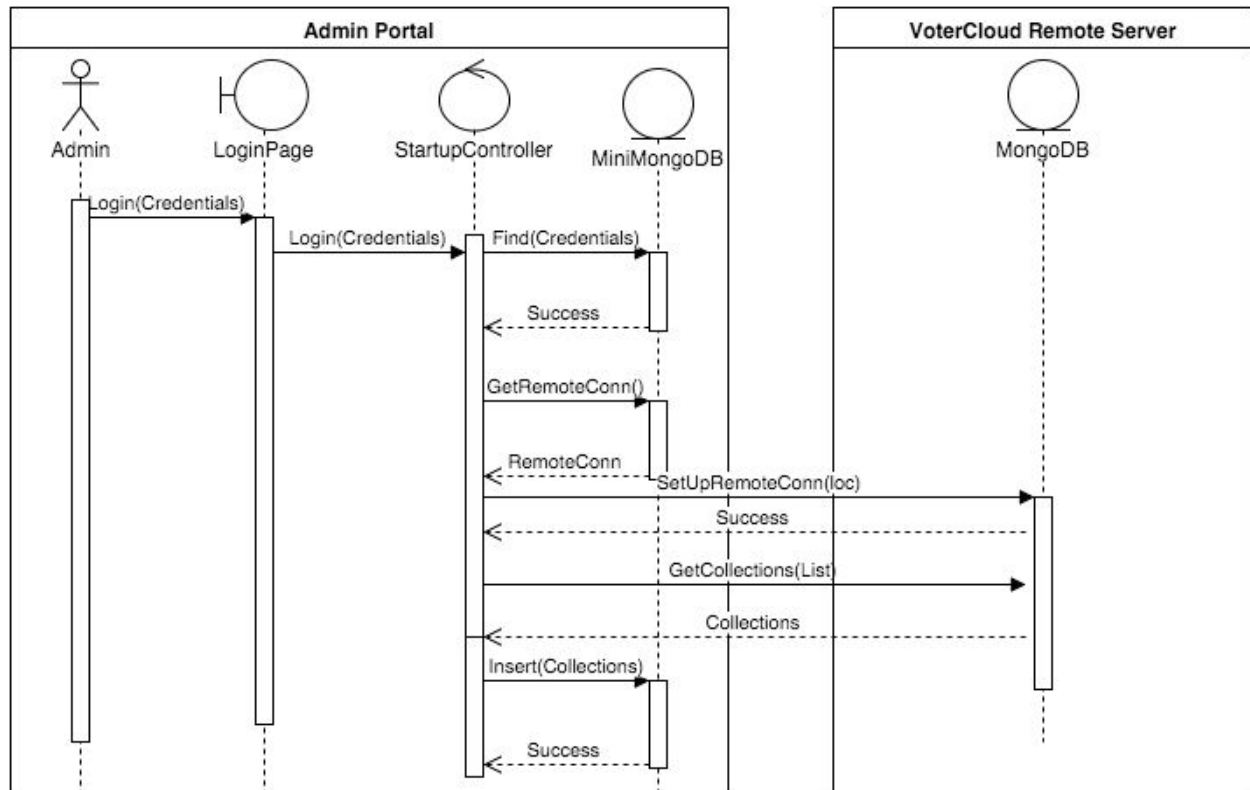
- As a VoterCloud administrator, I have to be able to make changes to the application such as posting Polls and setting general application environment variables.

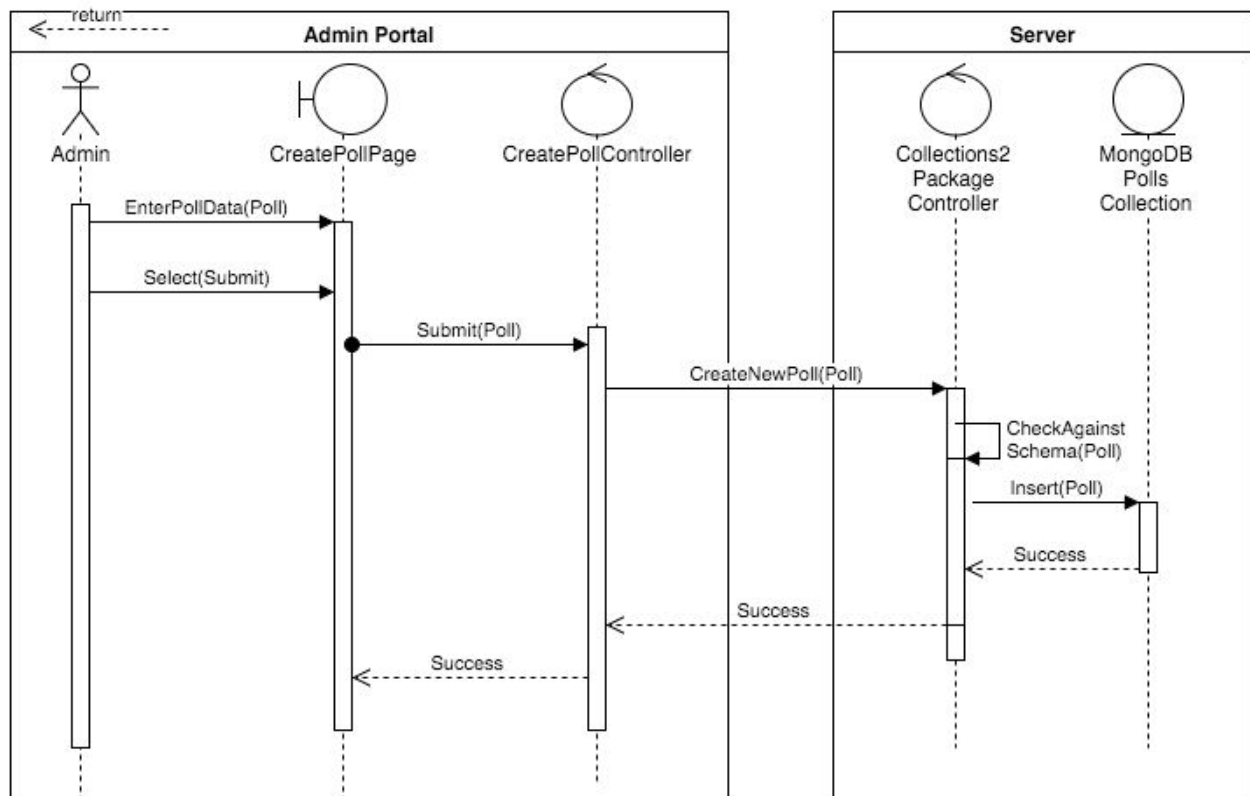
Acceptance Criteria:

1. Have a portal only accessible to the admins.
2. Make the necessary changes to the application by reading and writing to the application.
3. Shared Collections are reactive.
4. Validation controls are in place.



System Architecture with Respect to Admin





Admin Portal to create a Poll example.

User Story # 730- chat locality**Description:**

- As a user and engaged voter, i would like to communicate with other users based on region/location, so that i can engage based on my local information such as representatives and news.

Acceptance Criteria:

- location based chat.

2. unique channel for each chat created.
3. regions selection.

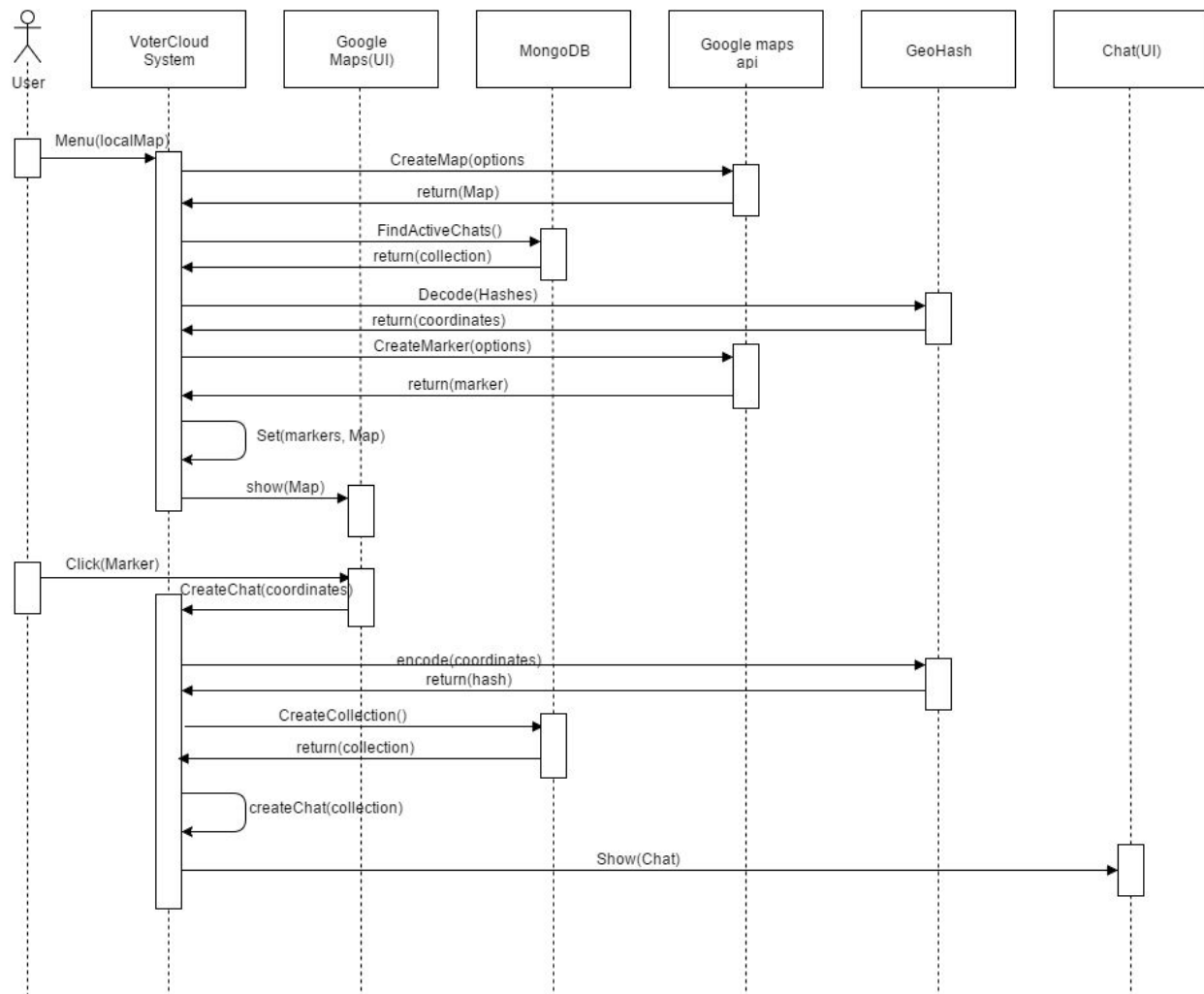
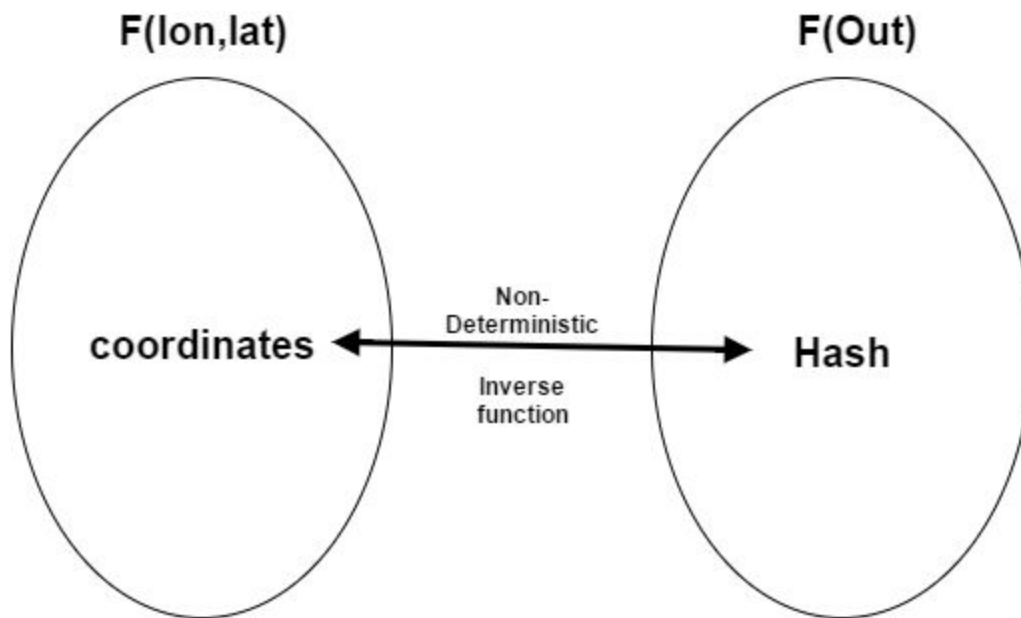


Figure 1.12 - The sequence diagram of the locality chat user story.

Main Packages: GeoHash - Take as input latitude and longitude and produce a hash code(that is inverse, able to be decode).



Map areas: implemented by the length of the geohash code, Such that if the length is 5 the rectangle size is of a zip code locality.

User Story # 736 representative chat

Description:

- as a user, i would like to engage on specific politician, so that i can chat with other about specific politician and create better engagement.

Acceptance Criteria:

1. md5 algorithm creates a unique and secure hash for each politician page.
2. unique page for each politician.
3. chat and description for each politician.

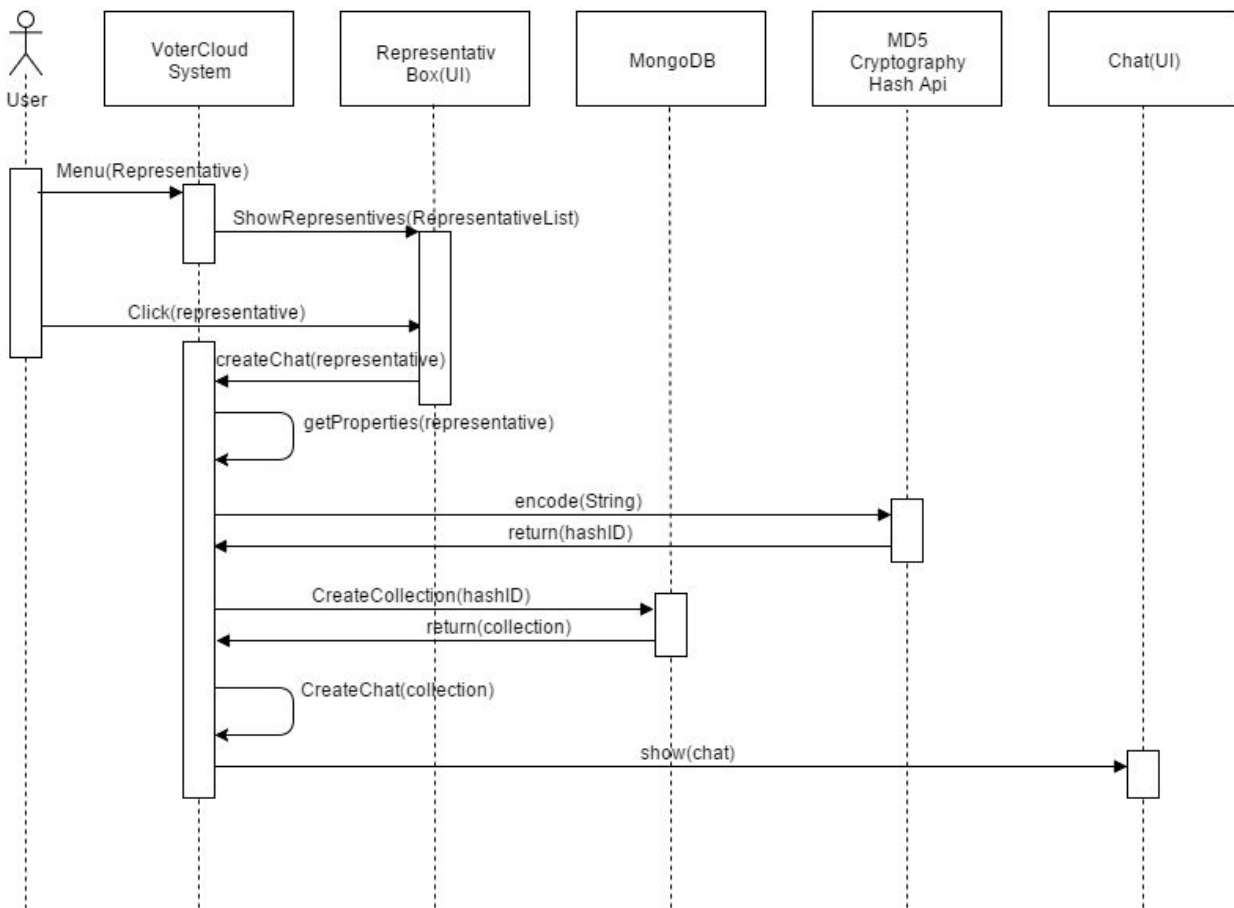


Figure 1.13 - The sequence diagram of the representative chat/page.

Reasons for using Hash function to implement the representative page:

1. the terms and conditions of the google civic api prohibit the developer to save the data in any kind.
2. secure and efficient use of routes and collections.

Criteria for the hash function:

1. Secured hash function (can not be decrypted).
2. efficient (generated fast).
3. Deterministic and non inverse (the same string produce the same result).

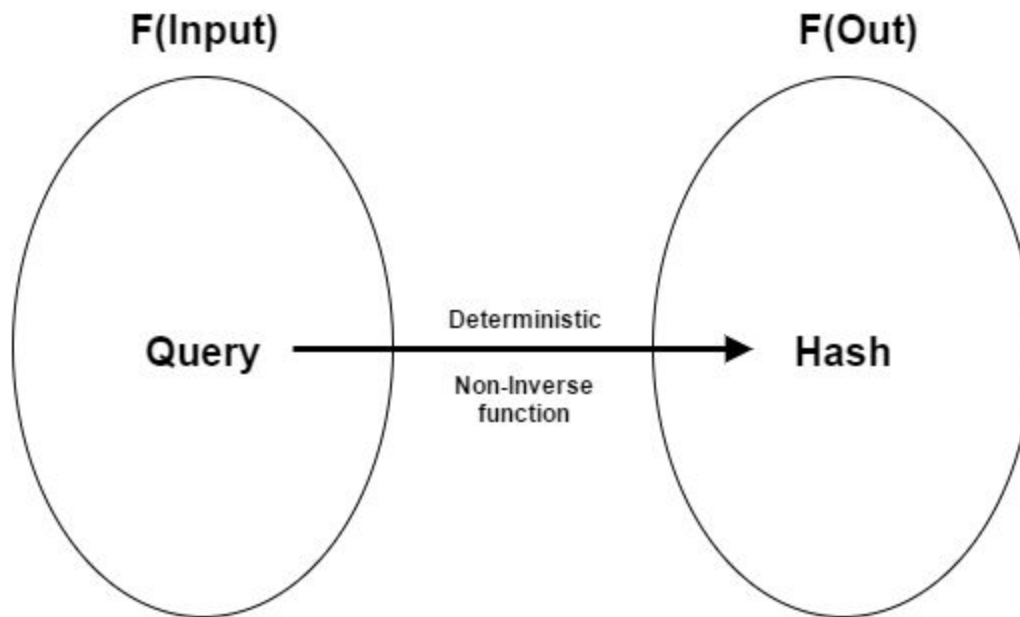


Figure 1.14 - hashing function description of criterias.

User Story # 729- Database security

Description:

- As a user and owner, i would like to make the mongoDB database collections secure, so that hackers can not manipulate the system and data.

Acceptance Criteria:

1. Publish/Subscribe methods(selections of what the client server see).
2. miniMongo Deny access to insert and update.

3. user can not manipulate the collections.

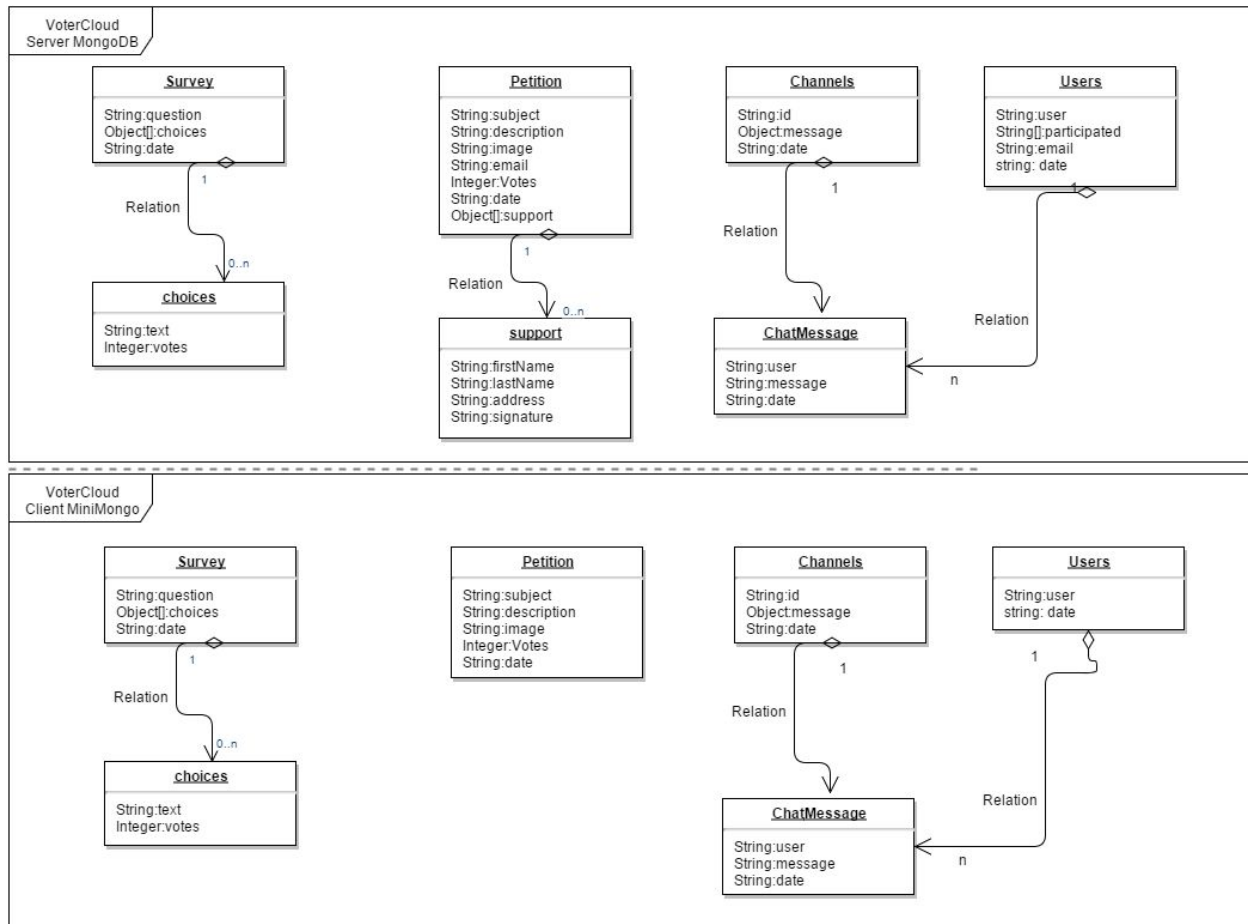


Figure 1.15 - Database security achieved by the idea of publish and subscribe on the client and server.

User Story # 721- Admin Portal

Description:

- As an owner, I want to add Polls and remove them, as well as review the database model so that I can create interaction for the users.

Acceptance Criteria:

4. Read all necessary collections.
5. Log in and logout successfully.
6. Write to necessary collections.

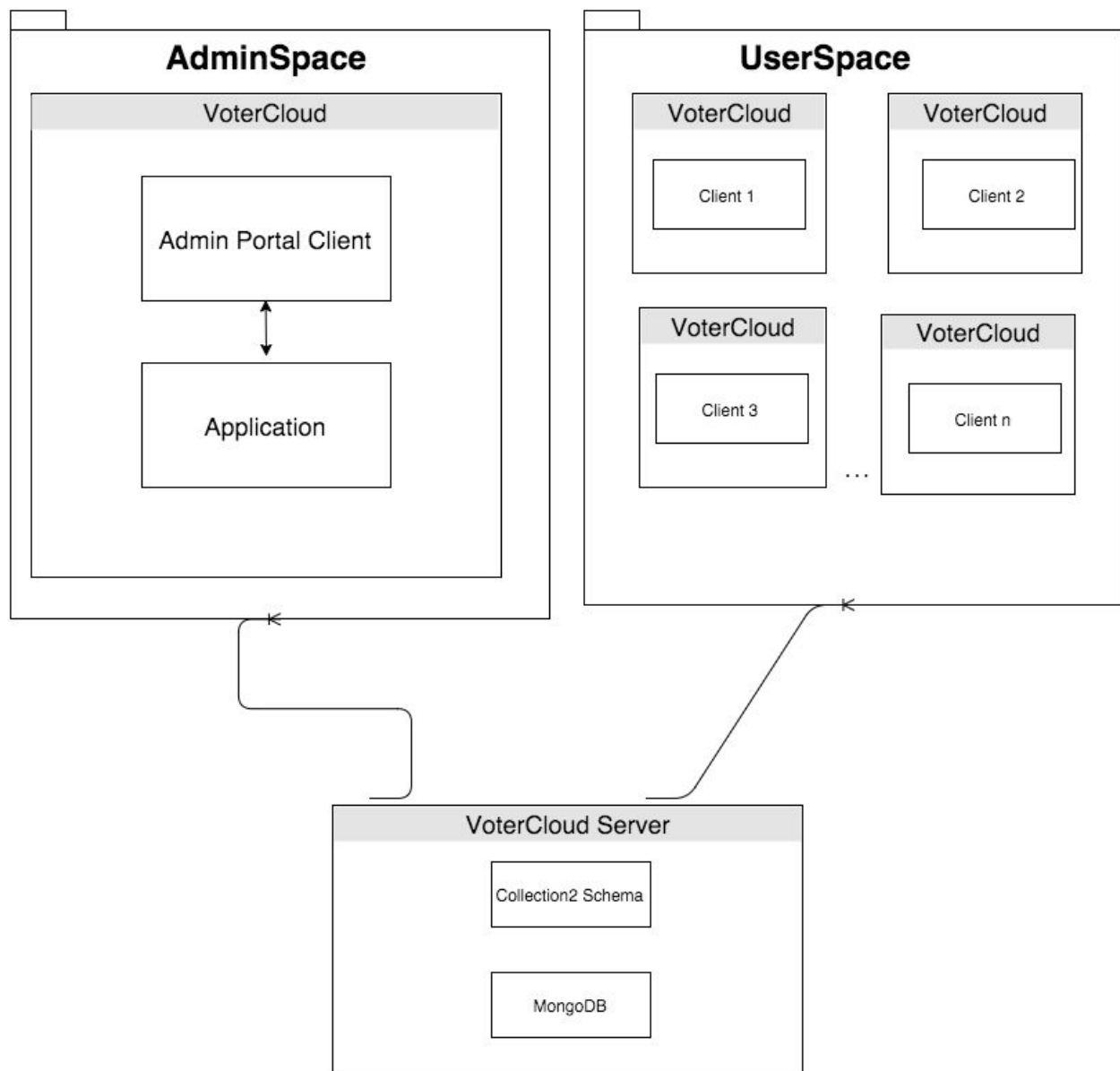


Figure 1.16 - Admin diagram, mapping between the subsystems of the portals.

Pending User Stories

- **Reliable Production Server**
 - Free Meteor servers are great for staging and testing. However, they are slow, spotty, and sometimes down. A production server like AWS would be a great solution.
- **Profile Page**
 - A user should have a profile page with social media features (profile picture, share with friends, etc.).
- **Voter ID # Access**
 - Somehow allow users to use their Voter ID # to access more features such as seeing if they are a lazy voter or not, historical voting information, etc.
- **Automatic Petition Email to Representative**
 - A petition should be tied to a Representative so that it can automatically be emailed once a proper number of users have been signed.
- **Features accessible to offline users**
 - Provide specific features to users not online. Like Representative search but possibly not chat.
- **Elections links stay within VoterCloud**
 - Links in Elections should open a window within the application so that the user never leaves the application and has a simple way to return from that link window.
- **Create Poll limited to Admin**
 - Limit the Create Poll to the Admin.
- **Improve Responsive Design**
 - Certain features and UI elements do not respond well to devices of all widths.
- **Allow Representatives to open a Representative Profile**
 - Representatives should be able to chat with the users on their page, provide updates, and possibly create polls.

PROJECT PLAN

The project plan is the overall planning and overview of how the project built and composed. As typical to agile development each day the team met and planned for the next day and reviewed the day in terms of results and implementations. And in the end of each two weeks we had a team/sprint review of what has been done during the last two weeks. The feedback enabled the team to make the proper and relevant changes so that it's match the product owner expectation. Each team member evaluated himself as well the other teammate to provide better team efficiency and orientation. The following paragraph describes exactly this methodology that we had followed. It's describes the milestones and the crucial component of our project as it's build ups (from sprint to sprint). It's includes the building stones of our system such as packages and libraries that our system use extensively. and the planning details and specification of each sprint and user stories for each sprint as it span throughout the sprint duration.

Hardware and Software Resources

DEVELOPMENT PLATFORM

- Meteor
 - Provides full stack capabilities to develop a mobile application or website with focus on reactive changes and developmental flexibility.
 - HTML5, CSS, Javascript

HOSTING

- Staging: votercloud-fiu.meteor.com
 - Using meteor.com's free hosting service.
 - Used to allow anyone to see the working application without installing it on their phones.
- Admin Portal:

SUPPORTED OPERATING SYSTEMS

- Android + iOS
 - Meteor allows for fast deployment of applications to Android and iOS devices.
 - Other operating systems not in focus due to testing and UI constraints.
- Web Browser

- FOR TESTING AND SHOWCASING PURPOSES ONLY.

METEOR PACKAGES

- meteor-platform
 - Standard Meteor pre-installed package
- insecure
 - Allow almost all collection methods, such as `insert`, `update`, and `remove`, to be called from the client.
- jquery
 - A javascript library designed to simplify javascript writing.
- iron:router
 - A page router that works on the server and the client.
- http
 - Allows application to make http calls to remote servers.
- sacha:spin
 - Spinner package for loading times.
- mdg:geolocation
 - Provides reactive geolocation on any device.
- meteorhacks:npm
 - Use npm modules with a Meteor app.
- accounts-password
 - A login service that enables secure password-based login
- accounts-ui
 - Simple template to add login widgets to an app.
- accounts-twitter
 - Login service for Twitter accounts
- twbs:bootstrap
 - Popular front-end framework for developing responsive, mobile-first projects.
- accounts-facebook
 - Login service for Facebook accounts.
- aldeed-Collection2
 - Automatic validation of insert and update operations on the client and server.
- aldeed-autoform
 - Takes your collection's schema and automatically create HTML5 forms based on it.
- steeve:jsignature

- jSignature is a jQuery plugin which simplifies creation of a signature capture field in the browser window, allowing a user to draw a signature using mouse, pen, or finger.
- meteorhacks:subs-manager
 - a general-purpose subscriptions manager for Meteor. It also works pretty well with Iron Router, with some limitations.
- meteorhacks:ssr
 - Server side rendering for Meteor with Blaze.
- dfischer:phantomjs
 - A scripted, headless browser used for automating web page interaction.

Sprints Plan

For each sprint, list the user stories selected for implementation in descending order of priority.

Sprint 1

(08/31/2015 - 09/11/2015)

User Story # 681 - Frameworks and Languages Tutorial

Tasks

- Learn Javascript (Raul and Eldar)
- Learn MongoDB (Raul and Eldar)
- Learn Meteor (Raul and Eldar)

Acceptance Criteria

- Javascript tutorial complete
- MongoDB tutorial complete
- Meteor tutorial complete

Modeling

Refer to UML diagrams in Appendix A that were created or modified to model the functionality that will be implemented in this sprint.

User Story # 684 - Introductory Portions of Documentation***Tasks***

- Edit Document Based on feedback
- Get feedback from mentor
- Write first page

Acceptance Criteria

- the introduction, current system, purpose of new system, and Architectural Patterns sections first draft is complete
- Each section has been reviewed by Gus, the mentor, and the sections have been edited to reflect his feedback.

Modeling

Refer to UML diagrams in Appendix A that were created or modified to model the functionality that will be implemented in this sprint.

User Story # 696 - Learn Amazon Web Service***Tasks***

- Research if AWS is a viable option

Acceptance Criteria

- Decision if AWS should be used in our project

Modeling

Refer to UML diagrams in Appendix A that were created or modified to model the functionality that will be implemented in this sprint.

User Story # 684 - Mockup UI***Tasks***

- Edit mockup using feedback
- Get feedback from mentor

- Design mockup

Acceptance Criteria

- complete design based on the vision and user stories.
- the design is clear and consistent.
- the mockup is basic and simple to follow.

Modeling

Refer to UML diagrams in Appendix A that were created or modified to model the functionality that will be implemented in this sprint.

Sprint 2

(09/11/2015 - 09/25/2015)

User Story # 677 - Menu***Tasks***

- List each task as a separate bullet point.
- The collection of tasks in a sprint defines its scope, i.e., the product to be delivered at the end of the sprint.
- The collection of tasks of all sprints defines the scope of the system.
- ...

Acceptance Criteria

- Menu can be reached from any page.
- Menu has list of all pages in the app.
- Menu can appear and disappear on command (unobtrusive).

Modeling

Refer to UML diagrams in Appendix A that were created or modified to model the functionality that will be implemented in this sprint.

User Story # 703 - Set up Meteor Hosting***Tasks***

- Research best Meteor hosting practices
- Set up hosting page

Acceptance Criteria

- Ability to deploy new code to meteor.com host
- Client code can communicate with server code in 2-way communication.

Modeling

Refer to UML diagrams in Appendix A that were created or modified to model the functionality that will be implemented in this sprint.

User Story # 672 - View Local Politicians***Tasks***

- Implement UI
- Write UML diagram
- Write Backend

Acceptance Criteria

- See a search result about the interest politician.
- See the corresponded politician profile and information.
- Be able to add the politician for the favorite list.

Modeling

Refer to UML diagrams in Appendix A that were created or modified to model the functionality that will be implemented in this sprint.

Sprint 3

(09/25/2015 - 10/09/2015)

User Story # 709 - Test Local Politicians***Tasks***

- Selenium testing

- Manual Testing
- Test Case

Acceptance Criteria

- All testing findings have been recorded.

Modeling

Refer to UML diagrams in Appendix A that were created or modified to model the functionality that will be implemented in this sprint.

User Story # 710 - Search By Location***Tasks***

- Search By IP Address
- Search By GPS

Acceptance Criteria

- Get correct latitude and longitude
- convert coordinates to real address.
- real time fast location search

Modeling

Refer to UML diagrams in Appendix A that were created or modified to model the functionality that will be implemented in this sprint.

User Story # 710 - About Information***Tasks***

- List team member name and relevant information.
- list role of each member.
- Name the product owner and mentor.
- UML

Acceptance Criteria

- About page is reachable and viewable as designed

Sprint 4
(10/09/2015 – 10/23/2015)

User Story # 715 - representative image profile***Tasks***

Number	Name	Status
720	representative image by twitter search api	Done
719	representative profile image by Wikipedia api	Done
718	representative image by twitter Id	Done
717	representative image by Facebook Id	Done

Acceptance Criteria

1. be able to see representative image profile from his facebook, twitter, wikipedia profile.
2. the image should be 150 px to 150 px match to the other images styles.
3. load time for calling the Facebook graph, Twitter, Wikipedia APIs should be minimize.

Modeling

Figure 1.05 - Sequence Diagram shows the sequence of interaction and actions for repr image.

User Story # 716 - about information***Tasks***

Number	Name	Status
747	reference	Done
746	team member info	Done

Acceptance Criteria

1. List team member name and relevant information.
2. list role of each member.
3. Name the product owner and mentor.

Modeling

No modeling, please look on the user interface section and the application to see how it's looks and presented.

User Story # 714 - selenium tests***Tasks***

Number	Name	Status
750	system test	Done
749	testing for the elections	Done
748	test representative	Done

Top of Form

Acceptance Criteria

1. all the features work properly.
2. no bug or errors.
3. fast running and smooth

Modeling

Please look on the testing and validation section to see the code and test cases.

Sprint 5

(10/23/2015 – 11/06/2015)

User Story # 675 - Surveys***Tasks***

Number	Name	Status
754	sort and order polls based on date added	Done
753	limit user participation	Done
752	create poll vetoing mechanize	Done
751	create poll page	Done

Acceptance Criteria

1. Forms can be filled.
2. The forms can be submitted.
3. the forms is added and collected as part of the statistical poll.

Modeling

Figure 1.06 - The sequence diagram of the survey user story.

User Story # 722 - petitions

Tasks

Number	Name	Status
759	limit user participation	Done
758	sort peitions	Done
757	make petition	Done
756	support peition	Done
755	petition page	Done

Acceptance Criteria

1. Allow any registered user to create a petition.

2. petition is dynamically updated.
3. Details of each user that signed.

Modeling

Figure 1.07 - The sequence diagram of the petition user story.

User Story # 723 - Signature

Tasks

Number	Name	Status
762	create a global variable	Done
761	save signature in efficient string format	Done
760	find sign package	Done

Acceptance Criteria

1. Be able to draw smoothly and signature.
2. Save the signature in compress and readable form.
3. Be able to store the image in MongoDB.

Modeling

Figure 1.11 - The sequence diagram of the signature user story.

User Story # 724 - Chat Room

Tasks

Number	Name	Status
766	create message format	Done
765	user identification on chat	Done
764	chat UI	Done
763	create chat mechanize	Done

Acceptance Criteria

1. Reactive chat.
2. Live feedback.
3. Be able to see the user and date of each message.

Modeling

Figure 1.08 - The sequence diagram of the chat user story.

User Story # 725 - PDF generator and serving***Tasks***

Number	Name	Status
770	research and find PDF helper package	Done
769	compile html on server sider	Done
768	decide on the PDF format and view	Done
767	render pdf into html	Done

Acceptance Criteria

1. render the adobe reader file.
2. the content of the PDF should be the total supportive of the petition.
3. should be presented only when the limit of supportive reached.

Modeling

Figure 1.09 - The sequence diagram of the pdf generator and serving after petition support.

Figure 1.10 - The sequence diagram of the pdf generator and serving after petition support.

Sprint 6

(11/06/2015 – 11/20/2015)

User Story # 729 - Database security

Tasks

Number	Name	Status
774	user separation and limitation on data	Done
773	database efficiency of what loaded	Done
772	server calls for data operation	Done
771	client server separation	Done

Acceptance Criteria

1. Publish/Subscribe methods(selections of what the client server see).
2. miniMongo Deny access to insert and update.
3. user can not manipulate the collections.

Modeling

Figure 1.15 - Database security achieved by the idea of publish and subscribe on the client and server.

User Story # 730 - chat locality**Tasks**

Number	Name	Status
735	google maps events and marks	Done
734	channel creation	Done
733	Google map	Done

Acceptance Criteria

1. location based chat.
2. unique channel for each chat created.
3. regions selection.

Modeling

Figure 1.12 - The sequence diagram of the locality chat user story.

User Story # 736 - representative chat***Tasks***

Number	Name	Status
775	research deterministic and non inverse hash	Done
737	hashing for each representative	Done

Acceptance Criteria

1. md5 algorithm creates a unique and secure hash for each politician page.
2. unique page for each politician..
3. chat and description for each politician.

Modeling

Figure 1.13 - The sequence diagram of the representative chat/page.

User Story # 721 - Admin portal***Tasks***

Number	Name	Status
728	Connect To Remote DB from Admin	

Acceptance Criteria

1. Special and authorize privileges.
2. Admin only access.
3. Easy login to the admin section with special and unique user.
4. Read all necessary collections.
5. Log in and logout successfully.
6. Write to necessary collections.

Modeling

Figure 1.16 - Admin diagram, mapping between the subsystems of the portals.

User Story # 731 - menu bar design

User Story # 738- ElectionsUI

User Story # 738- RepresentativesUI

Sprint 7

(11/20/2015 – 12/04/2015)

User Story # 726 - User Accounting***Tasks***

Number	Name	Status
667	login and logout	Done
666	Registration	Done

Acceptance Criteria

1. Be able to log in.
2. Be able to Register to the application.
3. Save and record user data correctly and completely.

Modeling

Please look on the user interface section for the images and appendix a for diagram.

User Story # 727- Create Hard Schema

Tasks

Number	Name	Status
776	create scheme design and policy	Done

Acceptance Criteria

1. Every data variable and command is accounted for in the schema.
2. Writing to and reading from the database works the same.
3. Validation controls are in place.

Modeling

the diagram of the database diagram presented in appendix a.

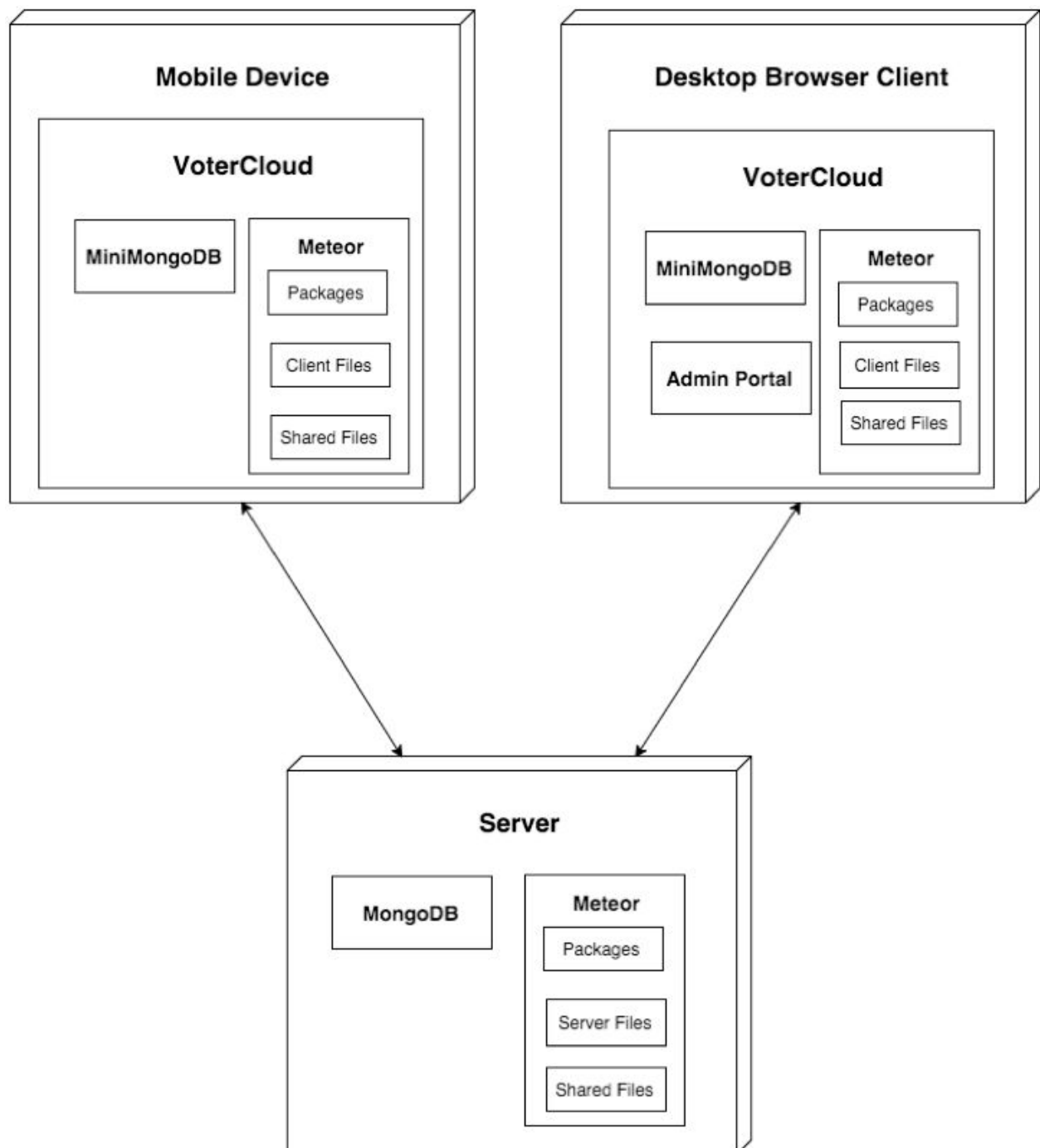
SYSTEM DESIGN

This section discuss the architecture pattern of the system and it's subsystem decomposition. The section makes up the complete description from the bottom up to how the system should behave and interact with the other subsystems. In terms of Architectural patterns, MVC describes perfectly the design and behavior of the system as discussed and described in great details in the next paragraph. The System and Subsystem decomposition describes the subsystems that makes our system as a whole, and their interaction with other subsystems. This paragraph is complete breakthrough of the system in terms of it's functionality and work. In addition the paragraph also describe and depicts the different relations among the subsystems and their interaction. The General System design of the application is waht called "isomorphic javascript" and can also be described as single-page app.

Hardware and Software Mapping

VoterCloud is a 3-part system. The primary portion is the Client side, which is in the form of a mobile application only with what is meant for the user to see (Client Files). It is accompanied by the central server that holds the primary database (MongoDB) and server logic (Server Files). The final portion is the Desktop Brower Client whose purpose is to provide

access to the Admin Portal. The Desktop Browser Client also includes the same files and programs as the Mobile Device version so that it can be viewed from the admin's browser.



Architectural Patterns

The system is an instance of isomorphic Javascript application, or in other words is the hybrid between the server side and the client side to create a single page app. That is communicated and works both in the client-side and server-side using the same algorithm and code.

Therefore the architecture pattern stay the same as regular client, server application. Though, this is special kind of MVC as depicted in the following figure:

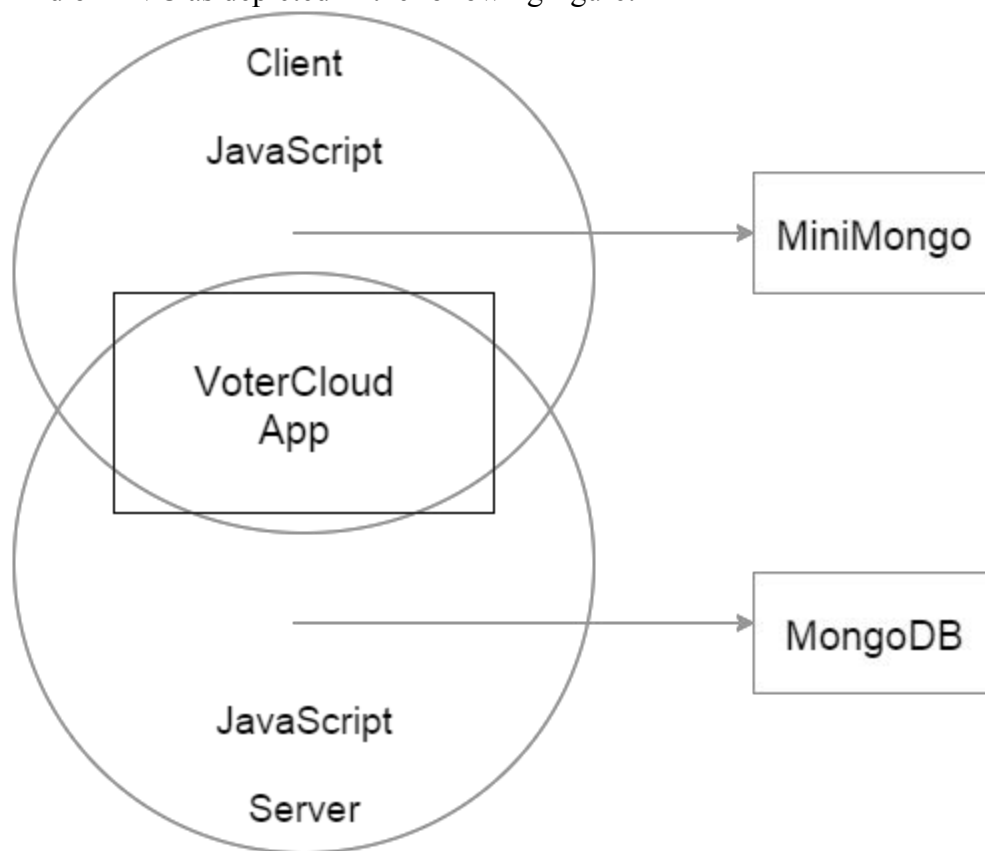


Figure 1.17 - MVC description of the client, server side.

The architecture pattern we identified is MVC (or Client-server MVC). because there is javascript running both on the server side and the client, and described as one layer, similarly to the client-serve MVC architecture. Therefore the application and the view logic can run both on the server and the client side. In addition the nature of Meteor.js describes the Model as MongoDB that is interconnected with the server and the client-side ("miniMongo").

As can be seen in figure 1.01 the MiniMongo resides in the client browser, while the MongoDB resides in the server side and contains full details and information of the data. Therefore using the nature of Meteor.js it's naturally fit to this description when describing one system doing both the view, control and model that are interconnected between the server side and client side using JavaScript.

System and Subsystem Decomposition

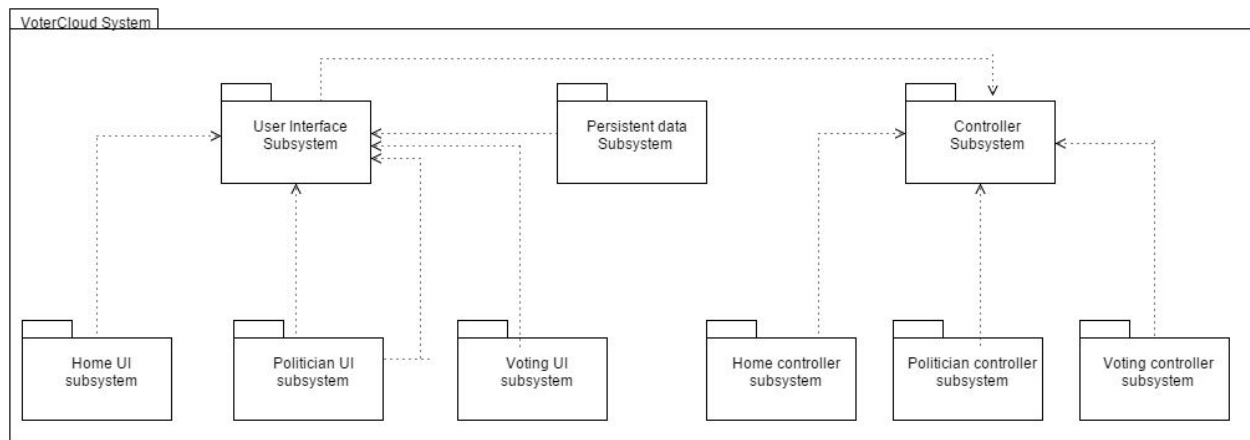


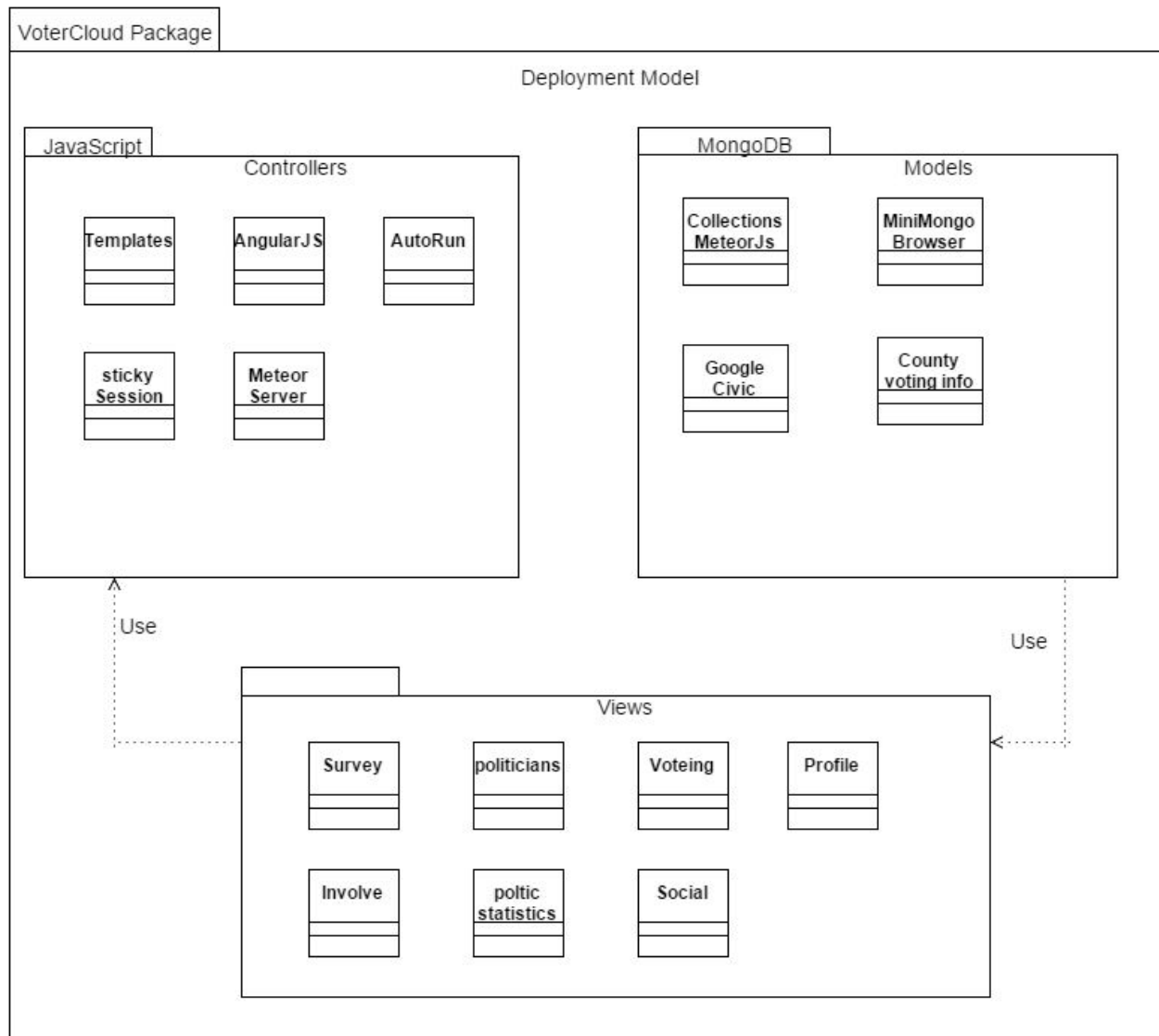
Figure 1.18 - Package diagram

- Persistent data - the model layer of the system. It's includes the MongoDB that resides on the server side and the MiniMongo that resides on the client/browser side. The subsystem Responsible for creating, retrieving, updating and manipulate records from the database. The jobs is accomplished by Meteor.js packages and architecture.
- User Interface - the view layer of the system. This layer consists with subsystems responsible for interacting with the user views.
 - Home UI - shows the available views to the user. In JavaScript terms it's called routing, that also deals with the permission and the related pages available to the user.
 - Politician UI - the politician administrator, shows the views available to the politician and politicians profile. It also provides the services and information of the politician.

- Voting UI - shows the available views related to the voting informations. It's primary dealing with the information presentation layer to the client, it's includes google civic api information.
- Controller- the control layer of the system. This layer consists with subsystems responsible for the logic and controlling the subsystems of the views layer.
 - Home Controller - is the main controlling subsystem, that is responsible for the routing and permission for different pages and services.
 - Politician Controller - the politician controller is responsible for receiving data from the database and the UI subsystem. It the main logic for the politician objects and clients.
 - Voting Controller - the voting controller is responsible for receiving data and using the google civic api to regulate and create the logic for the UI subsystem.

Deployment Diagram

The following figure describes the deployment diagram and the classes each one consists. It following the MVC architecture pattern with View, Model and Controller. They follows the general subsystems of the package diagram. In the view layer the system consists of the pages and views displayed to the client/user. The model layer uses MongoDB on the server side and MiniMongo on the client side. And the controller layer is responsible for the routing and the logic(it also includes the reactivity feature of meteor.js).

**Figure 1.19 - Deployment Model**

Design Patterns

As can also be seen from the class diagram the design patterns used in our project votercloud enabled us to create consistency and common interface of functionality. Therefore, much secure application and easier to understand and follows as it specify a clear object oriented relationships and functionalities. The design patterns that was used on this project are singleton, command and abstract factory.

- Singleton : we used this design patterns mostly for convenience and security, such that there is only one instance of each database entity and other functions require just to use and fetch this only entity. The benefits of doing so are to not create redundancy and confusion, it also enable us to create consistency throughout the code so that each object is inserted to his designated collection instance. Moreover, using this design pattern provide with great security and usability as each function and class can look on the collections as one interface and instance.
- Command : we used this design patterns to create common interface and convenience for commands that are frequently used. Such as set menu and set view, that effect the overall functionality and therefore crucial for making independent interface for the commands. The implementation of set view and menu follows this pattern so each function and class call this common interface/class to set the functionality of the command (from common function).
- Abstract Factory : this design patterns is perhaps the most useful and helpful in our project, since it's enabled us to implement more easily and faster the chat for each representative and map region chat. it's enabled us to create entities more easily and consistent in its design/functionality and implementation. In our project (votercloud) the abstract factory is designated for creating the chat interface and functionality for the representative and map location. So the chat functionality and characteristics are the same just the context of the chat is different.

SYSTEM VALIDATION

This paragraph describes the testing conducted to validate the system and the overall functionality of the system and the subsystems. The following user stories were tested using selenium, therefore this section also includes the code used to make the testing of each functionality and user story. This paragraph includes the test cases of each testing conducted for user story. The following test cases were first analyzed and then tested using automated testing (selenium) and manual testing, and compared to the expected result and actual result to check the matching and validation.

User Story # 730- chat locality

System Tests

- Test ID - The test was conducted to test the validation of the chat locality, or in general the google map functionality and correctness. this instance provide and describe the system functionality as a typical user click on the google map and tries to chat by on region and locality. Using Manual and Selenium automation testing.

Test Case ID	chat locality #730
Test Objective	Test that chat is created for each pointer click and pointer move(as typical user move to change location).
Precondition	<ol style="list-style-type: none">1. the gps is working, and produce the latitude and longitude and intended.2. the google map api is loaded.
Steps:	<ol style="list-style-type: none">1. Click on the menu button (to see the menu).2. Click on the chat locality..3. click on the red pointer to create a chat.4. click on the chat icon to join a chat.

	5. drag the red pointer to change location.
Test Data	message: "Hello representative "
Expected Result	the message and google maps functionality works as intended.
Actual Output	the message and google maps functionality works as intended.

User Story # 736- representative chat

System Tests

- Test ID - The test was conducted to test the validation of the chat for each representative functionality. this instance provide and describe the system functionality as a typical user click on representative and redirected to a new page that enable him to chat and see rep description. Using Manual and Selenium automation testing.

Test Case ID	representative chat #736
Test Objective	Test that the representative page in terms of redirecting and chat participation(checking chat for each representative).
Precondition	<ol style="list-style-type: none"> 1. the representative list is already loaded. 2. The lan and lat coordinates are enabled. 3. There are representative in the area.
Steps:	<ol style="list-style-type: none"> 1. Click on the menu button (to see the menu). 2. Click on the Representative page. 3. Click on search By GPS. 4. Click on the first representative. 5. add a comment to the chat.

Test Data	message: “Hello representative “
Expected Result	the message was added after the page was redirected.
Actual Output	the message was added after the page was redirected.

User Story # 722 - Petitions

System Tests

- Test ID - The test was conducted to test the validation of the petition support functionality. this instance provide and describe the system functionality as a typical user add and support some petition out of the petitions, by clicking on one of them and add the relevant input (first name, last name, address and signature). The end result of adding the input and support the petition is that the number of votes needed for this particular petition decremented by 1. Using Selenium automation testing.

Test Case ID	Petition #722
Test Objective	Test the Petition in terms of adding cast (support), and see if the input of the support added to the collection (MongoDB). The end result of this use case is seeing if the Votes needed decreased by 1.
Precondition	<ol style="list-style-type: none">1. The petition collections are loaded (MongoDB).2. The site is run and ready.3. Petition exists (there are petition to support).
Steps:	<ol style="list-style-type: none">1. Click on the menu button (to see the menu).2. Click on the Petition page.

	<ol style="list-style-type: none"> 3. Click on Support petition of the first petition. 4. Enter as input the first name, last name, address and signature. 5. Submit the form of this support petition.
Test Data	First Name: "Eldar" Last Name: "Feldbeine" Address: "Aventura, Florida" Signature: "point" Petition: "xpath of the first petition"
Expected Result	The petition Votes needed has been decremented successfully.
Actual Output	The petition Votes needed has been decremented successfully.

Code:
<pre> public static void main(String[] args) { WebDriver driver = new FirefoxDriver(); WebElement element=null; driver.get("http://votercloud-fiu.meteor.com/"); System.out.println("prepare for testing Petition User story"); try{ element=(new WebDriverWait(driver, 100)).until(ExpectedConditions.presenceOfElementLocated(By.id("image"))); } catch(StaleElementReferenceException ex){ System.out.println("No internet, or timeout loading"); } catch(RuntimeException ex) { System.out.println("no Internet"); } </pre>

```
    }
    driver.findElement(By.id("image")).click( );

    try{
        element = (new WebDriverWait(driver,
250)).until(ExpectedConditions.presenceOfElementLocated(By.linkText("Petition")));
    }
    catch(StaleElementReferenceException ex){
        System.out.println("No internet, or timeout loading");
    }

    driver.findElement(By.linkText("Petition")).click( );

    try {
        Thread.sleep(250);
    } catch (InterruptedException ex) {
        System.out.println("error");
    }

    driver.findElement(By.xpath("//*[@id=\"supportPetition\"]")).click( );

    try{
        element = (new WebDriverWait(driver,
250)).until(ExpectedConditions.presenceOfElementLocated(By.name("First")));
    }
    catch(StaleElementReferenceException ex){
        System.out.println("No internet, or timeout loading");
    }

    driver.findElement(By.name("First")).sendKeys("Eldar");
    driver.findElement(By.name("Last")).sendKeys("Feldbeine");
    driver.findElement(By.name("Address")).sendKeys("Aventura, Florida");
    driver.findElement(By.className("jSignature")).click();
    driver.findElement(By.id("Peti")).submit();

    System.out.println("Sprint 5, Checking the Petition user story");
    System.out.println("Successfully passed the selnium testing");
```

}

User Story # 675 - Survey**System Tests**

- Test ID - The test was conducted to test the validation of the survey interactivity. this instance provide and describe the system functionality as a typical user click on specific survey and see the number of votes incremented. Using Selenium automation testing.

Test Case ID	Surveys #675
Test Objective	Test the interactivity of the survey, by click on one of the options of the survey and see if it's incremented by exactly one (as result of a user click).
Precondition	<ol style="list-style-type: none"> 1. The survey collections are loaded (MongoDB). 2. The site is run and ready.
Steps:	<ol style="list-style-type: none"> 1. Click on the menu button (to see the menu). 2. Click on the Survey page. 3. Click on the first choice of the first survey.
Test Data	Message: survey 1 (xpath).
Expected Result	The survey has been incremented successfully.
Actual Output	The survey has been incremented successfully.

Code:

```
public static void main(String[] args)
{
    WebDriver driver = new FirefoxDriver();
    WebElement element=null;
    driver.get("http://votercloud-fiu.meteor.com/");
    System.out.println("prepare for testing Survey User story");
    try{
        element=(new WebDriverWait(driver,
100)).until(ExpectedConditions.presenceOfElementLocated(By.id("image")));
    }
    catch(StaleElementReferenceException ex){
        System.out.println("No internet, or timeout loading");
    }
    catch(RuntimeException ex)
    {
        System.out.println("no Internet1");
    }
    driver.findElement(By.id("image")).click( );

    try{
        element = (new WebDriverWait(driver,
250)).until(ExpectedConditions.presenceOfElementLocated(By.linkText("Survey")));
    }
    catch(StaleElementReferenceException ex){
        System.out.println("No internet, or timeout loading");
    }

    driver.findElement(By.linkText("Survey")).click( );

    try {
        Thread.sleep(250);
    } catch (InterruptedException ex) {
        System.out.println("error");
    }

    driver.findElement(By.xpath("//*[@id=\"choice\"]/span[2]")).click( );
```

```

System.out.println("Sprint 5, Checking the Survey user story");
System.out.println("Successfully passed the selenium testing");
}

```

User Story # 724 - Chat room

System Tests

- Test ID - The test was conducted to test the validation of the send message on the chat room using the dashboard or the input box. this instance provide and describe the system functionality as a typical user click and load input to the input box to be send to the collection MongoDB and presented that on the board. Using Selenium automation testing.

Test Case ID	Chat Room #724
Test Objective	Test that a message can be send and presented on the board of messages.
Precondition	<ol style="list-style-type: none"> 1. The message collection is loaded (MongoDB). 2. The site is run and ready.
Steps:	<ol style="list-style-type: none"> 1. Click on the “msg” input box. 2. Add the desire message (as input). 3. Submit the message (Click enter or send).
Test Data	Message: "Message TEST !!!!!"
Expected Result	The message has been sent successfully.
Actual Output	The message has been sent successfully.

Code:

```
public static void main(String[] args)
```

```
{
    WebDriver driver = new FirefoxDriver();
    WebElement element=null;
    driver.get("http://localhost:3000/");
    System.out.println("prepare for testing Chat");
    try{
        element=(new WebDriverWait(driver,
100)).until(ExpectedConditions.presenceOfElementLocated(By.id("msg")));
    }
    catch(StaleElementReferenceException ex){
        System.out.println("No internet, or timeout loading");
    }
    catch(RuntimeException ex)
    {
        System.out.println("no Internet1");
    }
    driver.findElement(By.id("msg")).sendKeys("Message TEST !!!!!");

    try{
        element = (new WebDriverWait(driver,
250)).until(ExpectedConditions.presenceOfElementLocated(By.id("sendMsg")));
    }
    catch(StaleElementReferenceException ex){
        System.out.println("No internet, or timeout loading");
    }
    }

    driver.findElement(By.id("sendMsg")).submit();

    System.out.println("Sprint 5, Checking the Chat room user story");
    System.out.println("Successfully passed the selenium testing");
    }
```

User Story # 710 - Search by location

System Tests

- Test ID - The test was conducted to test the validation of the search by location of the representatives. this instance provide and describe the system functionality as a typical user click on the gps search button option. Using Selenium automation testing.

Test Case ID	Search by location # 710
Test Objective	Test the search representatives by location (GPS or IP address).
Precondition	<ol style="list-style-type: none"> 1. The site is loaded and ready. 2. The JavaScript startup function loaded the relevant locations based on GPS and IP address.
Steps:	<ol style="list-style-type: none"> 1. Click on the menu button (to see the menu). 2. Click on the Search page. 3. Click on Search by GPS. 4. Wait for the API calls and loading.
Test Data	IP Address: "your IP" GPS coordinates: "GPS latitude and longitude"
Expected Result	The representatives are loaded successfully.
Actual Output	The representatives are loaded successfully.

```

public static void main(String[] args)
{
    WebDriver driver = new FirefoxDriver();
    WebElement element=null;
    driver.get("http://votercloud-fiu.meteor.com/");
    System.out.println("prepare for testing");
    try{
        element=(new WebDriverWait(driver,

```

```
100)).until(ExpectedConditions.presenceOfElementLocated(By.id("image")));
    }
    catch(StaleElementReferenceException ex){
        System.out.println("No internet, or timeout loading");
    }
    catch(RuntimeException ex)
    {
        System.out.println("no Internet1");
    }
    driver.findElement(By.id("image")).click( );

    try{
        element = (new WebDriverWait(driver,
250)).until(ExpectedConditions.presenceOfElementLocated(By.linkText("Representatives")));
    }
    catch(StaleElementReferenceException ex){
        System.out.println("No internet, or timeout loading");
    }

    driver.findElement(By.linkText("Representatives")).click( );

    try{
        element = (new WebDriverWait(driver,
250)).until(ExpectedConditions.presenceOfElementLocated(By.name("but")));
    }
    catch(StaleElementReferenceException ex){
        System.out.println("No internet, or timeout loading");
    }
    try {
        Thread.sleep(250);
    } catch (InterruptedException ex) {
        System.out.println("error");
    }
    driver.findElement(By.name("but")).click();

    try{
        element = (new WebDriverWait(driver,
```

```

250)).until(ExpectedConditions.presenceOfElementLocated(By.id("officialName")));
    }
    catch(StaleElementReferenceException ex){
        System.out.println("No internet, or timeout loading");
    }

    System.out.println("Sprint 3, Checking the Representative functionality by GPS/IP
address");
    System.out.println("Successfully passed the selnium testing");
}

```

User Story # 672 - View Local politicians

System Tests

- Test ID - The test was conducted to test the validation of the search representatives by typed address. this instance provide and describe the system functionality as a typical user input a requested address to find representatives based upon that address. Using Selenium automation testing.

Test Case ID	View Local politicians# 672
Test Objective	Test the search representatives typing an input address directly, using the input box.
Precondition	<ol style="list-style-type: none"> 1. The site is loaded and ready. 2. The JavaScript startup function loaded the relevant locations.
Steps:	<ol style="list-style-type: none"> 1. Click on the menu button (to see the menu). 2. Click on the Search page. 3. Type input Address into the Input box. 4. Wait for the API calls and loading to see list of representatives.

Test Data	Address: “Aventura, Florida”
Expected Result	The representatives are loaded successfully.
Actual Output	The representatives are loaded successfully.

```

public static void main(String[] args)
{
    WebDriver driver = new FirefoxDriver();
    WebElement element=null;
    driver.get("http://votercloud-fiu.meteor.com/");
    System.out.println("prepare for testing");
    try{
        element=(new WebDriverWait(driver,
100)).until(ExpectedConditions.presenceOfElementLocated(By.id("image")));
    }
    catch(StaleElementReferenceException ex){
        System.out.println("No internet, or timeout loading");
    }
    catch(RuntimeException ex)
    {
        System.out.println("no Internet1");
    }
    driver.findElement(By.id("image")).click( );

    try{
        element = (new WebDriverWait(driver,
250)).until(ExpectedConditions.presenceOfElementLocated(By.linkText("Representatives")));
    }
    catch(StaleElementReferenceException ex){
        System.out.println("No internet, or timeout loading");
    }

    driver.findElement(By.linkText("Representatives")).click( );

```

```
try{
    element = (new WebDriverWait(driver,
250)).until(ExpectedConditions.presenceOfElementLocated(By.name("but")));
}
catch(StaleElementReferenceException ex){
    System.out.println("No internet, or timeout loading");
}
try {
    Thread.sleep(250);
} catch (InterruptedException ex) {
    System.out.println("error");
}

driver.findElement(By.name("address")).sendKeys("11200 SW 8th St, Miami, FL
33199");
driver.findElement(By.id("Search")).submit();
try{
    element = (new WebDriverWait(driver,
250)).until(ExpectedConditions.presenceOfElementLocated(By.id("officialName")));
}
catch(StaleElementReferenceException ex){
    System.out.println("No internet, or timeout loading");
}

System.out.println("Sprint 2, Checking the Representative functionality");
System.out.println("Successfully passed the selenium testing");
}
```

User Story # 713 - Elections

System Tests

- Test ID - The test was conducted to test the validation of the election user story to see the upcoming elections and their details(when you click on them). this instance provide and describe the system functionality as a typical user access to the election/voting and the detail information of each election. Using Selenium automation testing.

Test Case ID	Elections # 713
Test Objective	Test the election page in terms of seeing the upcoming elections and information of each specific election.
Precondition	<ol style="list-style-type: none"> 1. The site is loaded and ready. 2. The JavaScript startup function loaded the relevant functions (google civic api).
Steps:	<ol style="list-style-type: none"> 1. Click on the menu button (to see the menu). 2. Click on the Election page. 3. Click on one of the upcoming election. 4. Test and verify that the election information appeared as the result of the click.
Test Data	None
Expected Result	The Upcoming Election and the election information are loaded successfully and correctly.
Actual Output	The Upcoming Election and the election information are loaded successfully and correctly.

```

public static void main(String[] args)
{
    WebDriver driver = new FirefoxDriver();
    WebElement element=null;
    driver.get("http://votercloud-fiu.meteor.com/");
    System.out.println("prepare for testing");
}

```

```
try{
    element=(new WebDriverWait(driver,
100)).until(ExpectedConditions.presenceOfElementLocated(By.id("image")));
}
catch(StaleElementReferenceException ex){
    System.out.println("No internet, or timeout loading");
}
catch(RuntimeException ex)
{
    System.out.println("no Internet1");
}
driver.findElement(By.id("image")).click( );

try{
    element = (new WebDriverWait(driver,
250)).until(ExpectedConditions.presenceOfElementLocated(By.linkText("Elections")));
}
catch(StaleElementReferenceException ex){
    System.out.println("No internet, or timeout loading");
}

driver.findElement(By.linkText("Elections")).click( );

try{
    element = (new WebDriverWait(driver,
250)).until(ExpectedConditions.presenceOfElementLocated(By.linkText("North Carolina
Municipal General Election")));
}
catch(StaleElementReferenceException ex){
    System.out.println("No internet, or timeout loading");
}
try {
    Thread.sleep(250);
} catch (InterruptedException ex) {
    System.out.println("error");
}
```

```

        driver.findElement(By.linkText("North Carolina Municipal General Election")).click();
        try{
            element = (new WebDriverWait(driver,
250)).until(ExpectedConditions.presenceOfElementLocated(By.id("officialName")));
        }
        catch(StaleElementReferenceException ex){
            System.out.println("No internet, or timeout loading");
        }

        System.out.println("Sprint 3, Checking the Elections, more specifically the North Carolina
Municipal General Election");
        System.out.println("Successfully passed the selenium testing");
    }

```

User Story # 739- Admin

System Tests

- Test ID - The test was conducted to test the validation of the Admin page for reading, writing, and deleting to any the collections in MongoDB. The admin user should be able to login in, select the collection he or she wants to see. chat for each representative functionality.

Test Case ID	Admin-Read
Test Objective	Test that all the collections can be read properly.
Precondition	<ol style="list-style-type: none"> 1. Collections exists 2. The collections are preloaded and known
Steps:	<ol style="list-style-type: none"> 1. Add a row to each collection. 2. See it is reflected in the Admin Portal 3. Remove a row from each collection 4. See if it reflected in the Admin Portal
Test Data	Posts.remove({_id : 3}) Posts.insert((new Post(...)))

Expected Result	The removed post cannot be seen from the Admin Portal anymore The inserted post can be seen from the Admin Portal.
Actual Output	The removed post could not be seen from the Admin Portal anymore The inserted post was seen from the Admin Portal.

Test Case ID	Admin-Write
Test Objective	Test that a collection could be written to.
Precondition	1. A collection exists. 2. The collection is viewable from the Admin Portal
Steps:	1. Select the “Add Post” option. 2. Fill in the fields. 3. Submit Post.
Test Data	Question: “Can this Admin post be seen?” Choice1: “Yes” Choice2: “No”
Expected Result	The post is submitted and added to the collection and can be viewed from the application.
Actual Output	The newly inserted post breaks the code because it inserts the “choices” as an Object rather than an array.

User Story - Registration to the VoterCloud accounting System.

System Tests

- Test ID - The test was conducted to test the validation of the registration and login to the system. this instance provide and describe the system functionality as a typical user attempt to login or register to the system using typical input. The testing was conducted manually and with the help of Selenium automation testing.

Test Case ID	Registration
Test Objective	Test and verify that the registration works, such that in registration a user is added under unique registration to the system.
Precondition	<ol style="list-style-type: none"> 1. The site is loaded and ready. 2. The Meteor.users database is loaded correctly. 3. the server is in persistent connection with the user.
Steps:	<ol style="list-style-type: none"> 1. Click on register. 2. set the Test Data(Input). 3. Click Submit. 4. Test and verify the the Votercloud application is loaded as it supposed to be.
Test Data	UserName - EldarFIU Passwords - EldarFIU Zipcode - 33180 Email - Eldar@gmail.com
Expected Result	The Site is loaded into the main Votercloud application.
Actual Output	The Site is loaded into the main Votercloud application.

User Story - Login to the VoterCloud accounting System.

System Tests

- Test ID - The test was conducted to test the validation of the registration and login to the system. this instance provide and describe the system functionality as a typical user attempt to login or register to the system using typical input. The testing was conducted manually and with the help of Selenium automation testing.

Test Case ID	Login
Test Objective	Test and verify that the login works, such that under login an existing user is logged in to the system.
Precondition	<ol style="list-style-type: none">1. The site is loaded and ready.2. The Meteor.users database is loaded correctly.3. the server is in persistent connection with the user.
Steps:	<ol style="list-style-type: none">1. Click on login.2. set the Test Data(Input).3. Click Submit.4. Test and verify the the Votercloud application is loaded as it supposed to be.
Test Data	UserName - EldarFIU Passwords - EldarFIU
Expected Result	The Site is loaded into the main Votercloud application.
Actual Output	The Site is loaded into the main Votercloud application.

MOBILE APPLICATION TESTING

The following is a table of the testing done of VoterCloud when it has been pushed as a mobile application. The test cases are all that have been done in the previous section with the exception of the admin portal since it is not accessible through the mobile application, only through a web browser.

Meteor provides the functionality to deploy a project to a mobile application through the command *meteor run android-device* for an Android device.

Device: Samsung Galaxy S6

Operating System: Android 5.1.1

Test Case ID	WiFi Result	4G LTE Result
chat locality #730	Map does not load	Map does not load
representative chat #736	Representative photo does not load	Representative photo does not load
Petition #722	Petition photo does not load	Petition photo does not load
Surveys #675	Success	Success
Chat Room #724	User photos do not load	User photos do not load
Search by location # 710	Success	Success
Local politicians# 672	Representatives' photos does not load	Representatives' photos does not load
Elections #713	Success	Success
Registration	Success	Success
Login	Success	Success

GLOSSARY

Admin	The VoterCloud Product Owner or anyone who he has granted access to the Admin Portal to make changes
Channel	Akin to a chat room.
Supporter	A person signing a petition.

APPENDIX

Appendix A - UML Diagrams

Static UML Diagrams

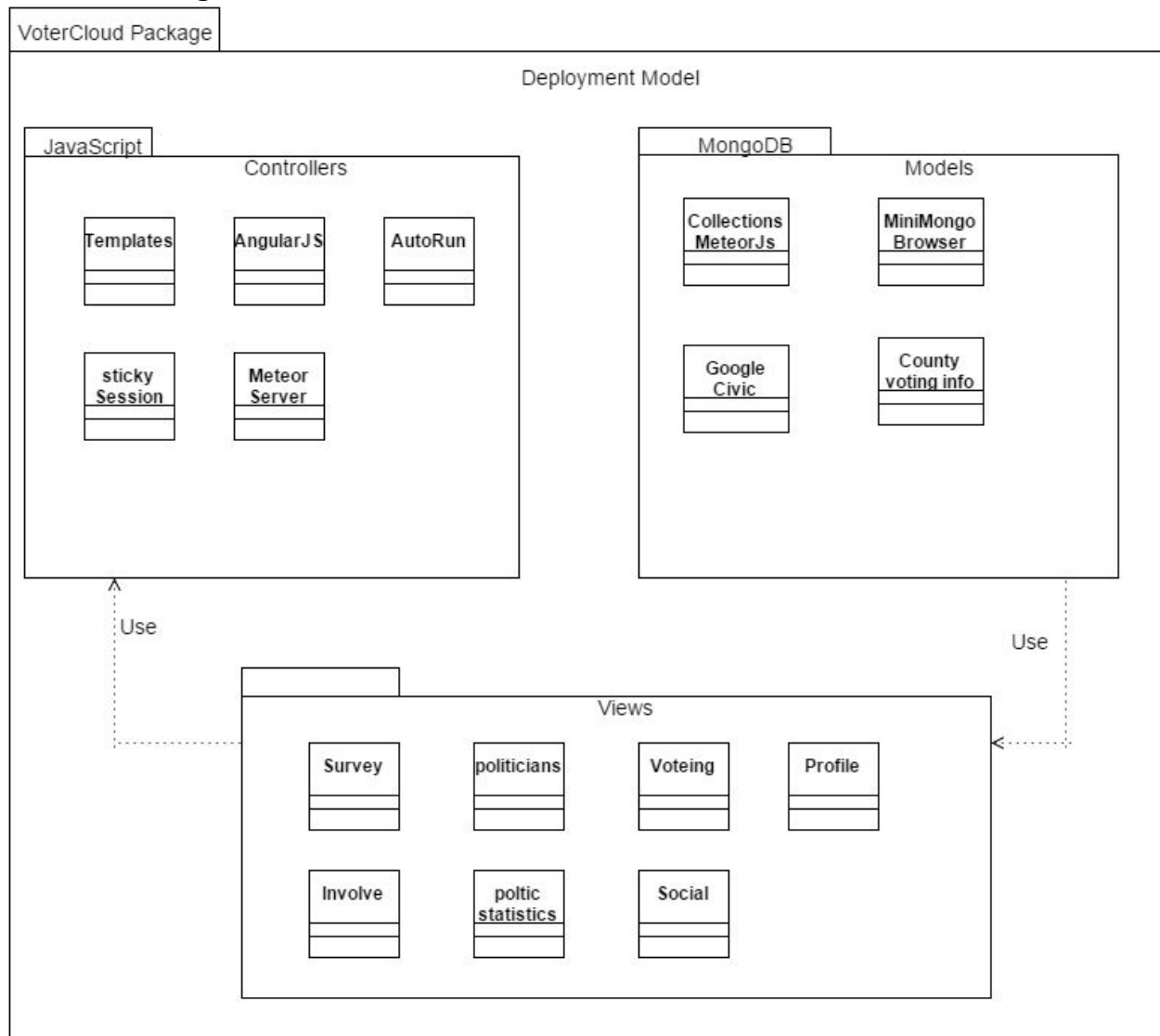


Figure 1.01 - Deployment Model

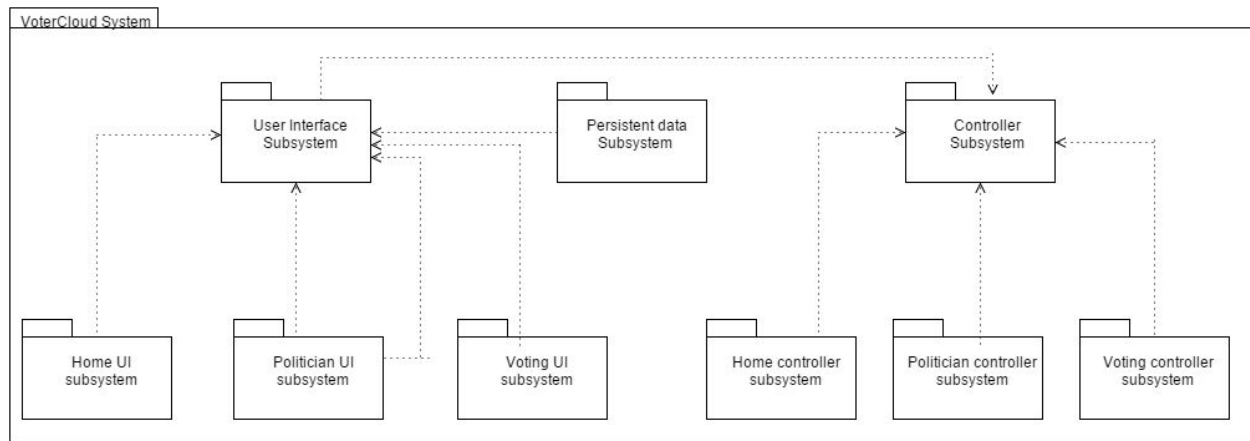


Figure 1.02 - Package diagram

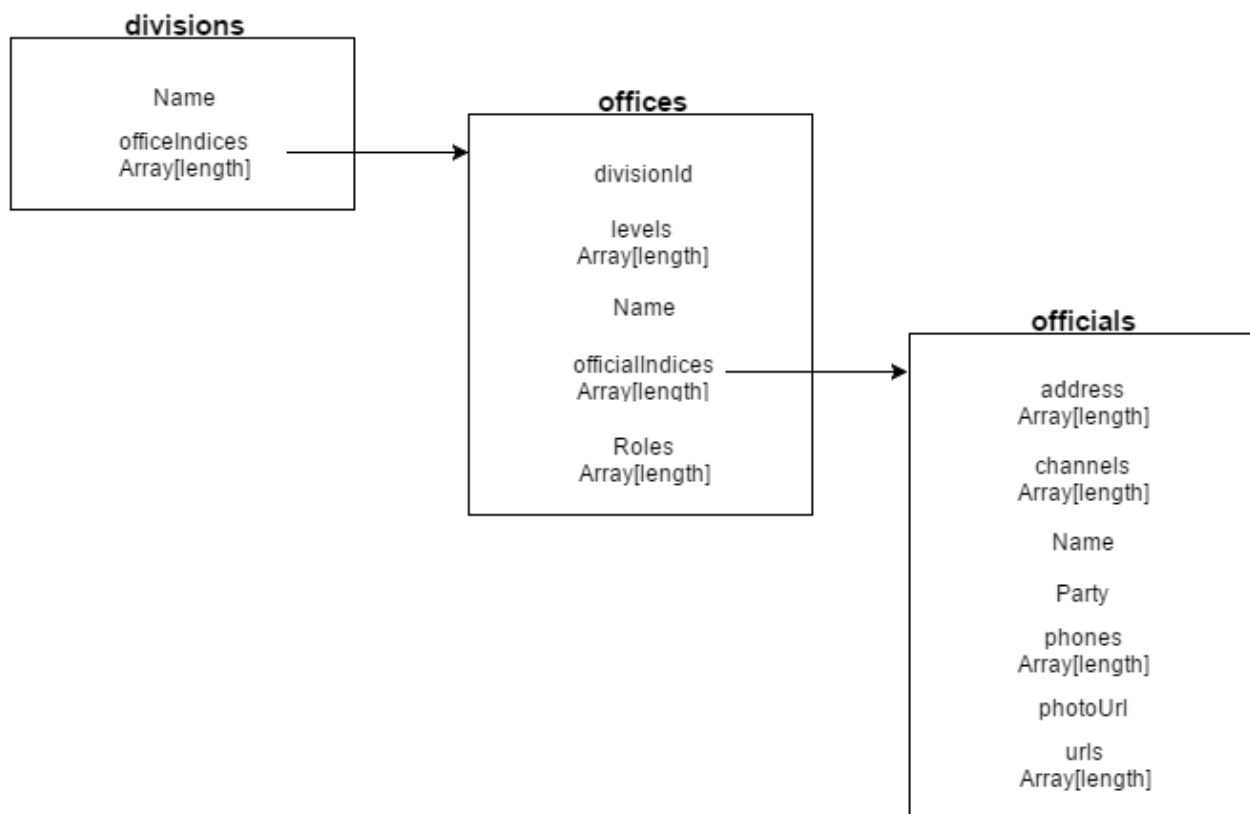
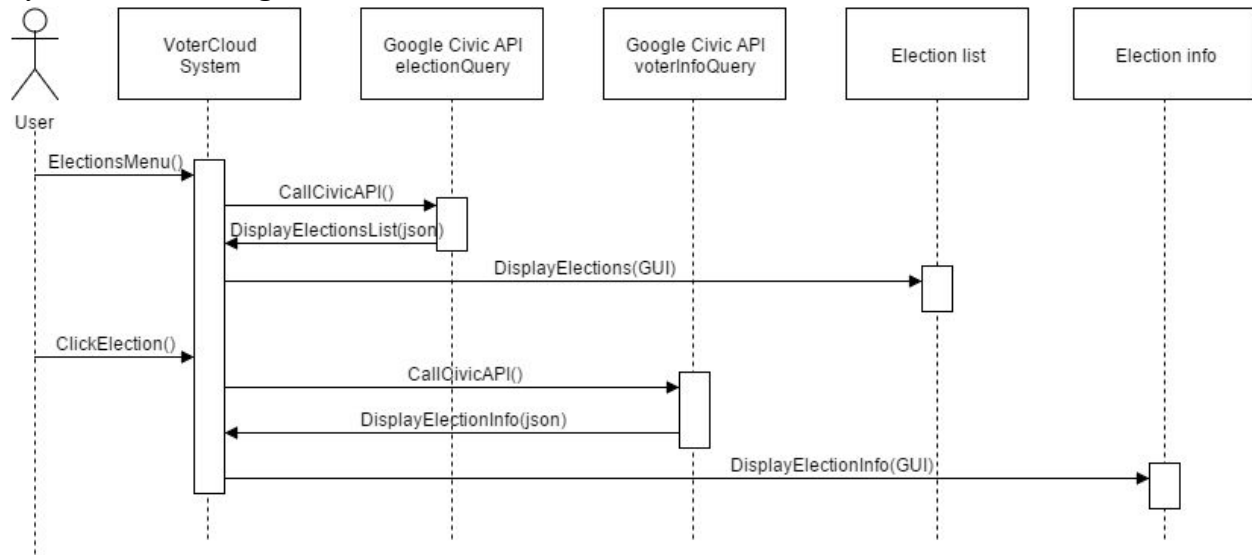


Figure 1.03 - Json file result of google civic api.

Dynamic UML Diagrams**Figure 1.03** - Sequence diagram of the election user story.

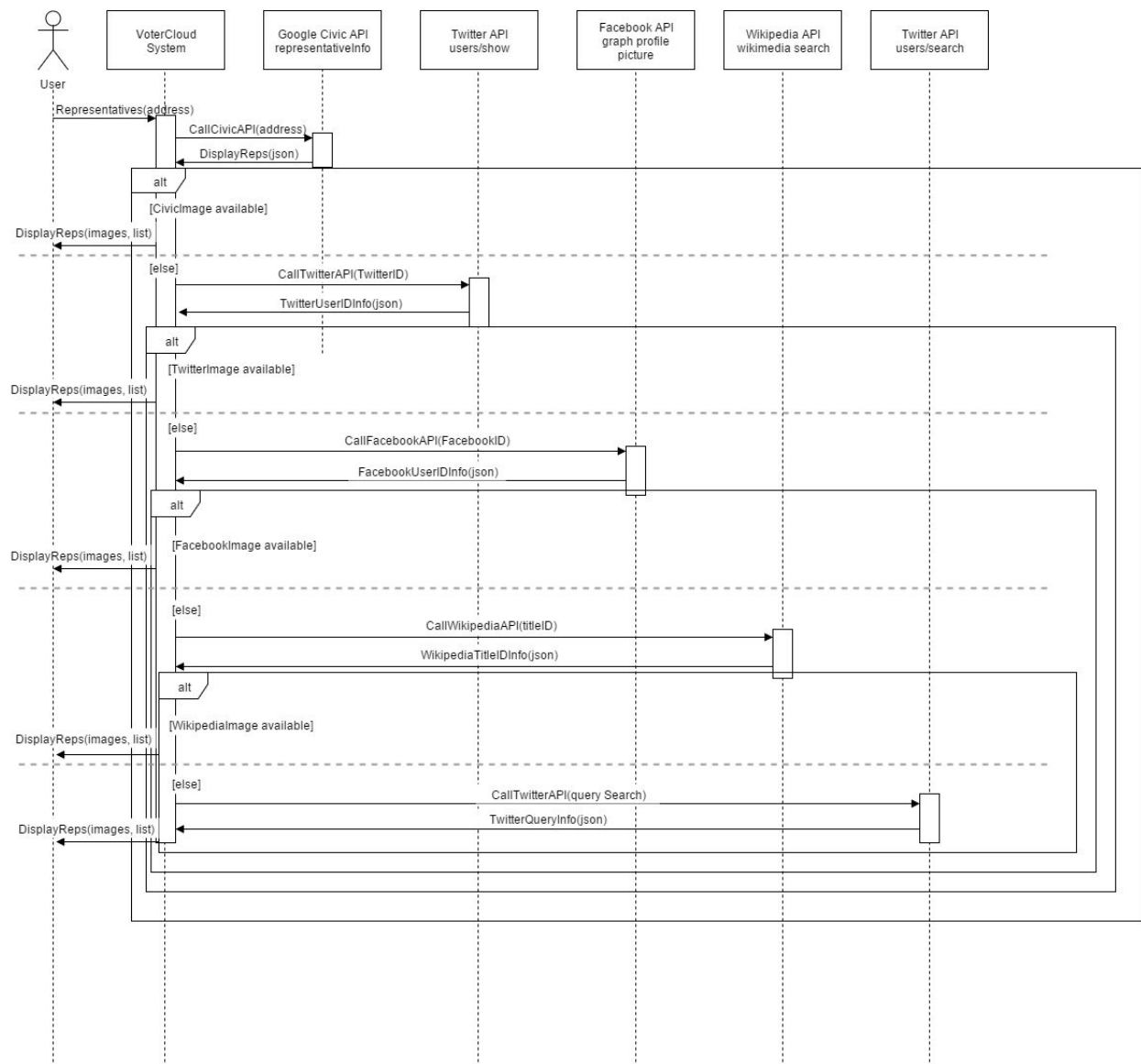


Figure 1.05 - Sequence Diagram shows the sequence of interaction and actions for repr image.

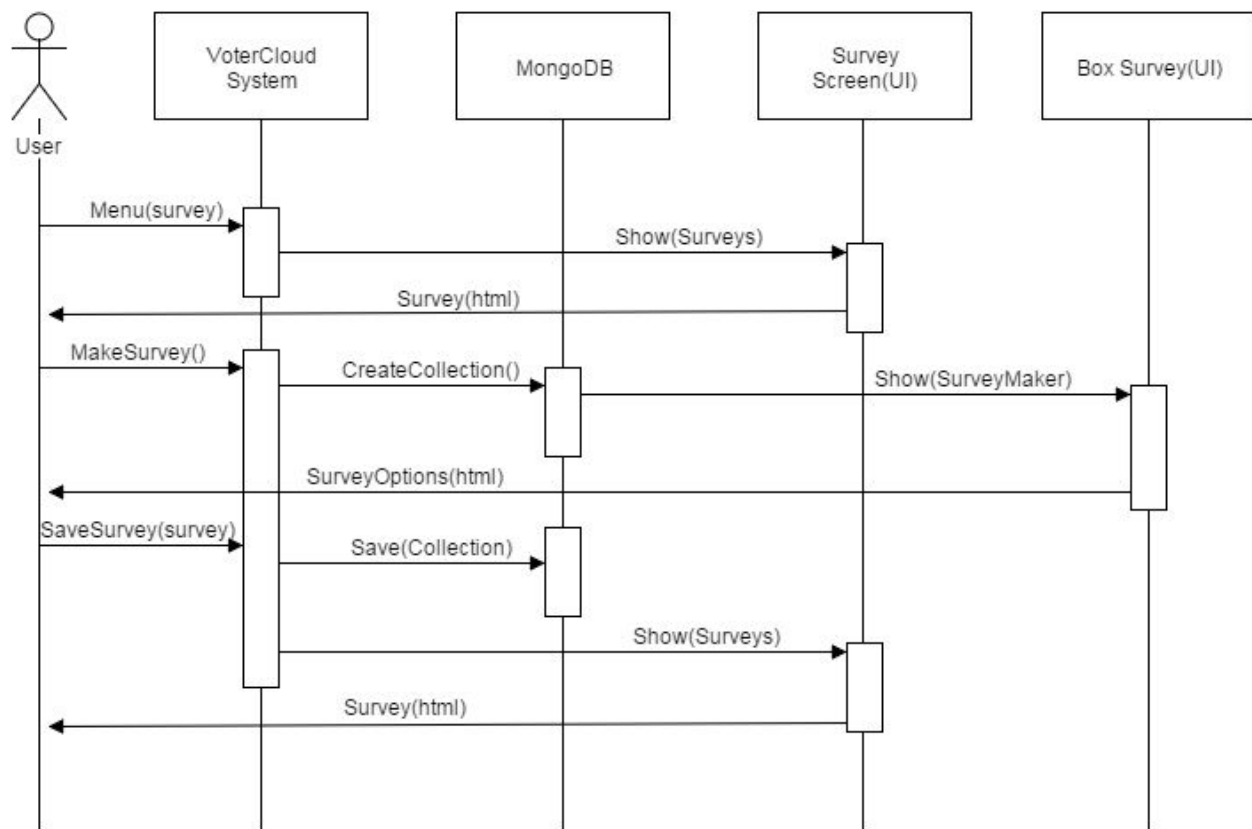


Figure 1.06 - The sequence diagram of the survey user story.

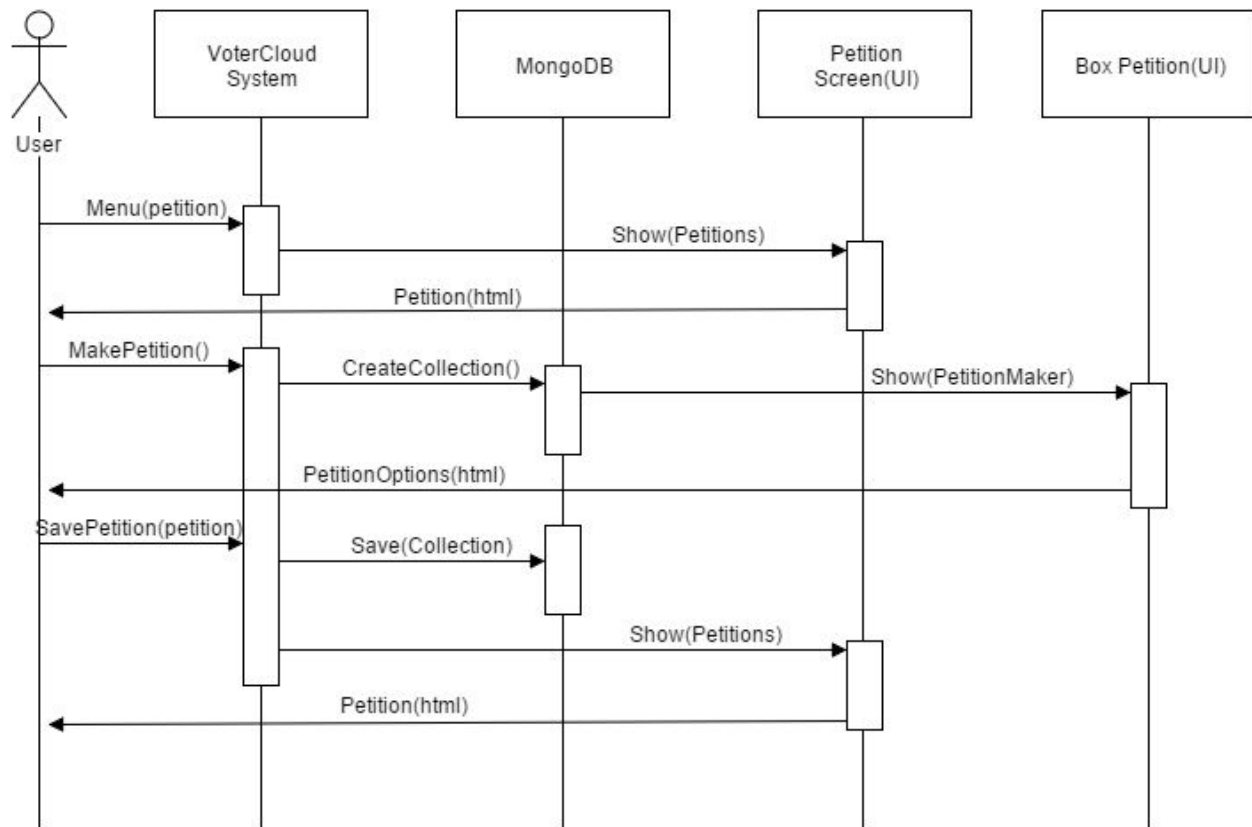


Figure 1.07 - The sequence diagram of the petition user story.

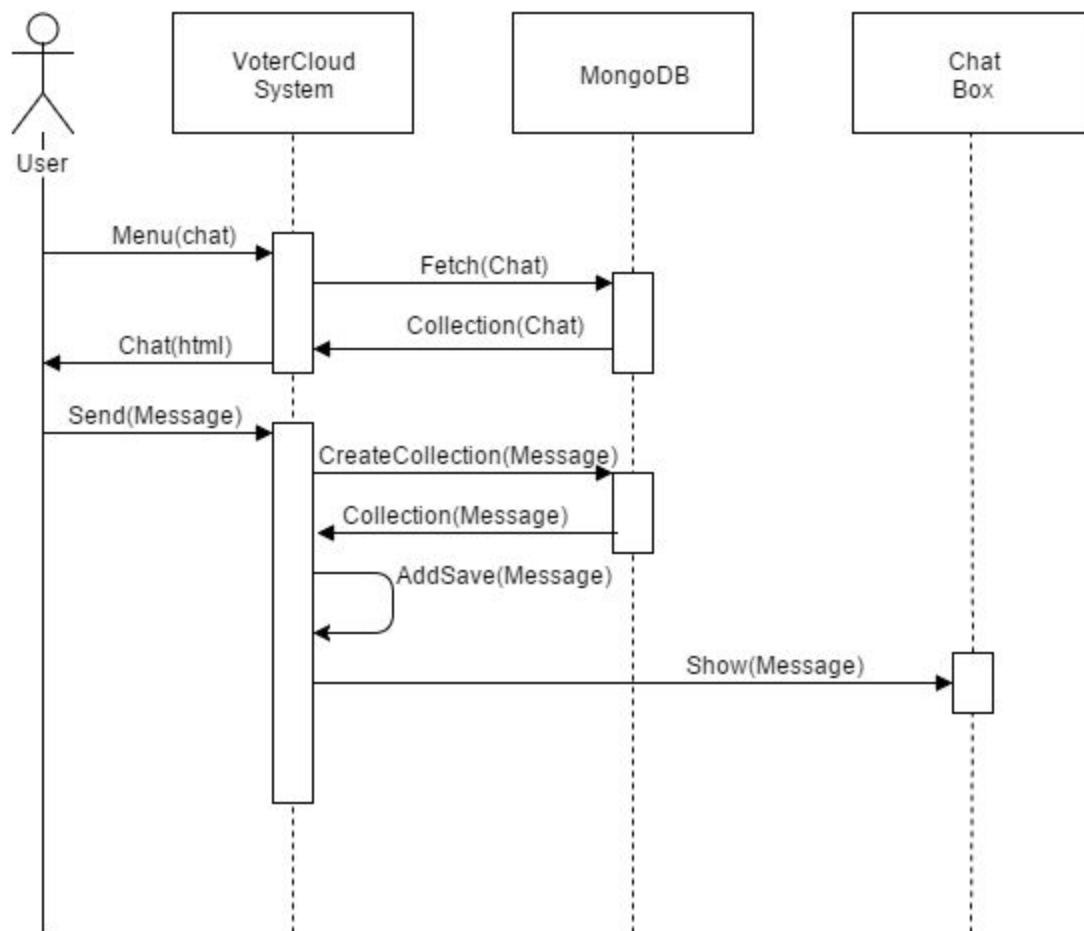


Figure 1.08 - The sequence diagram of the chat user story.

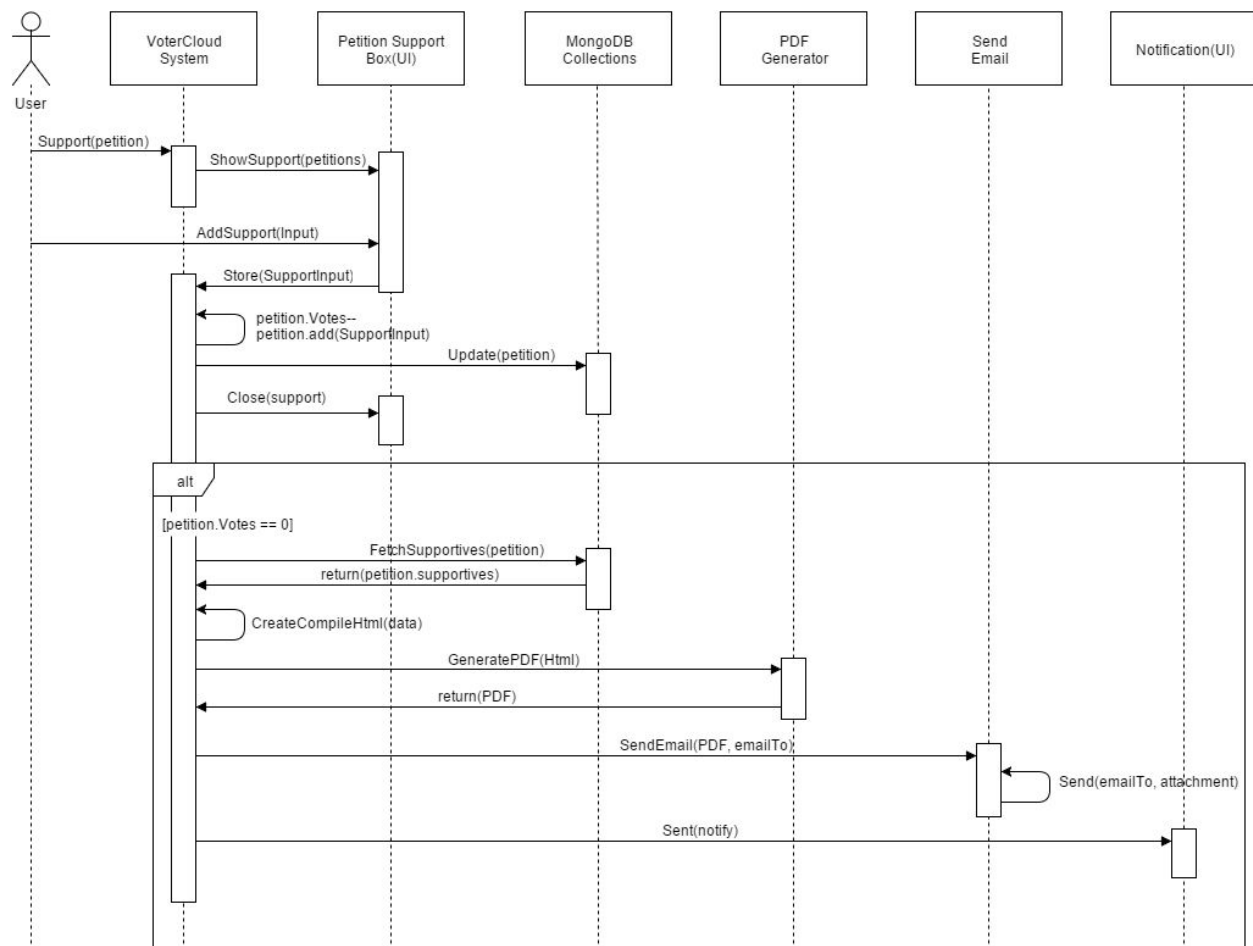


Figure 1.09 - The sequence diagram of the pdf generator and serving after petition support.

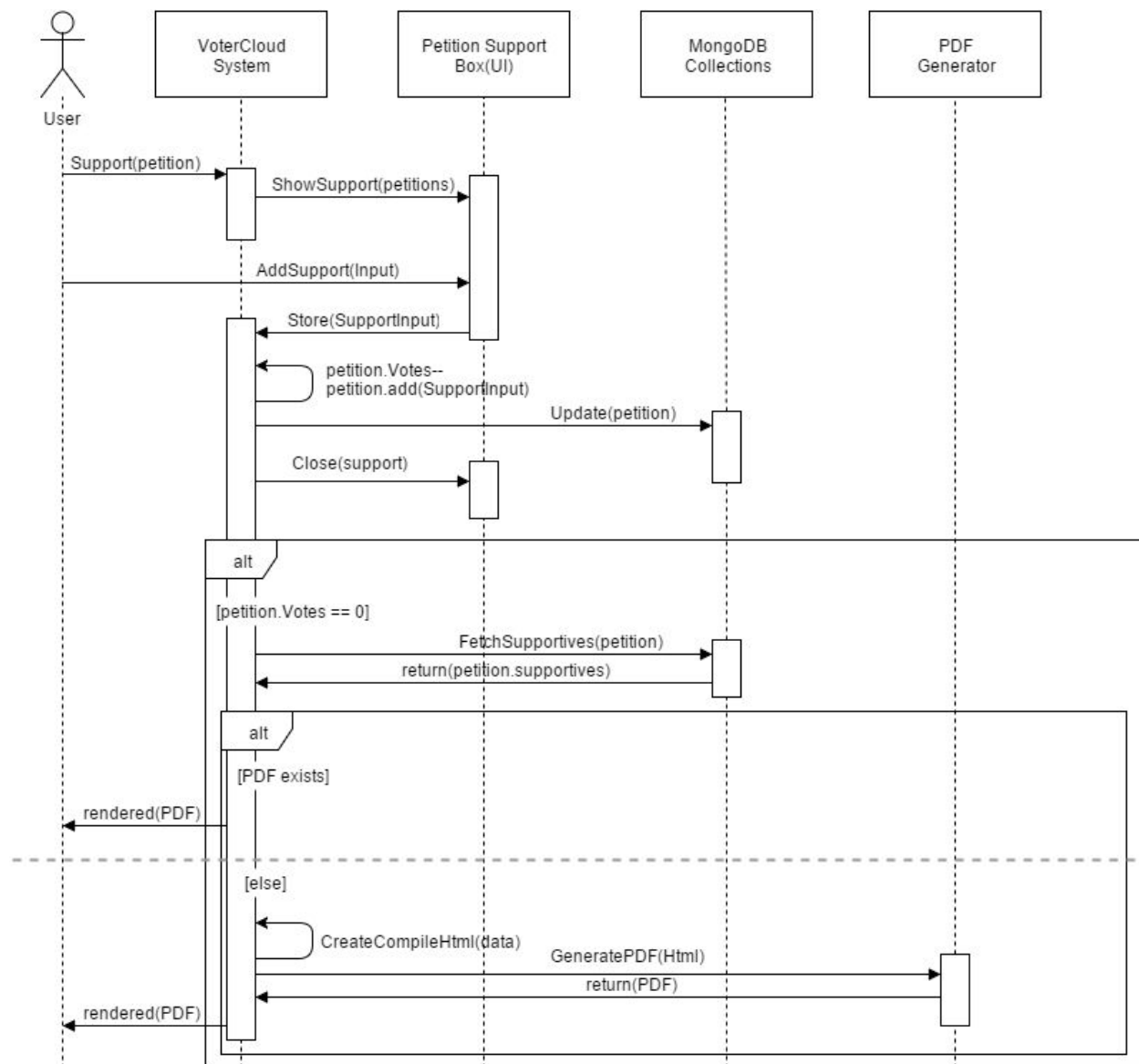


Figure 1.10 - The sequence diagram of the pdf generator and serving after petition support.

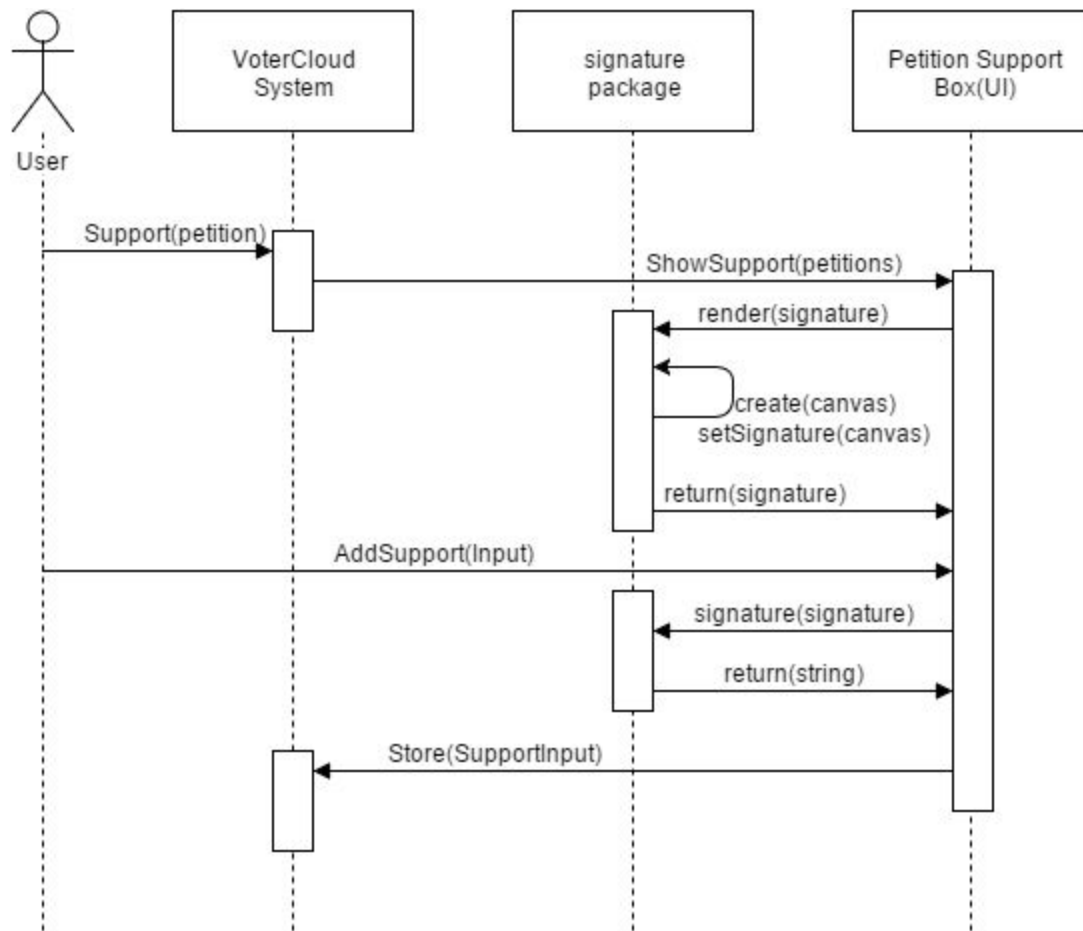


Figure 1.11 - The sequence diagram of the signature user story.

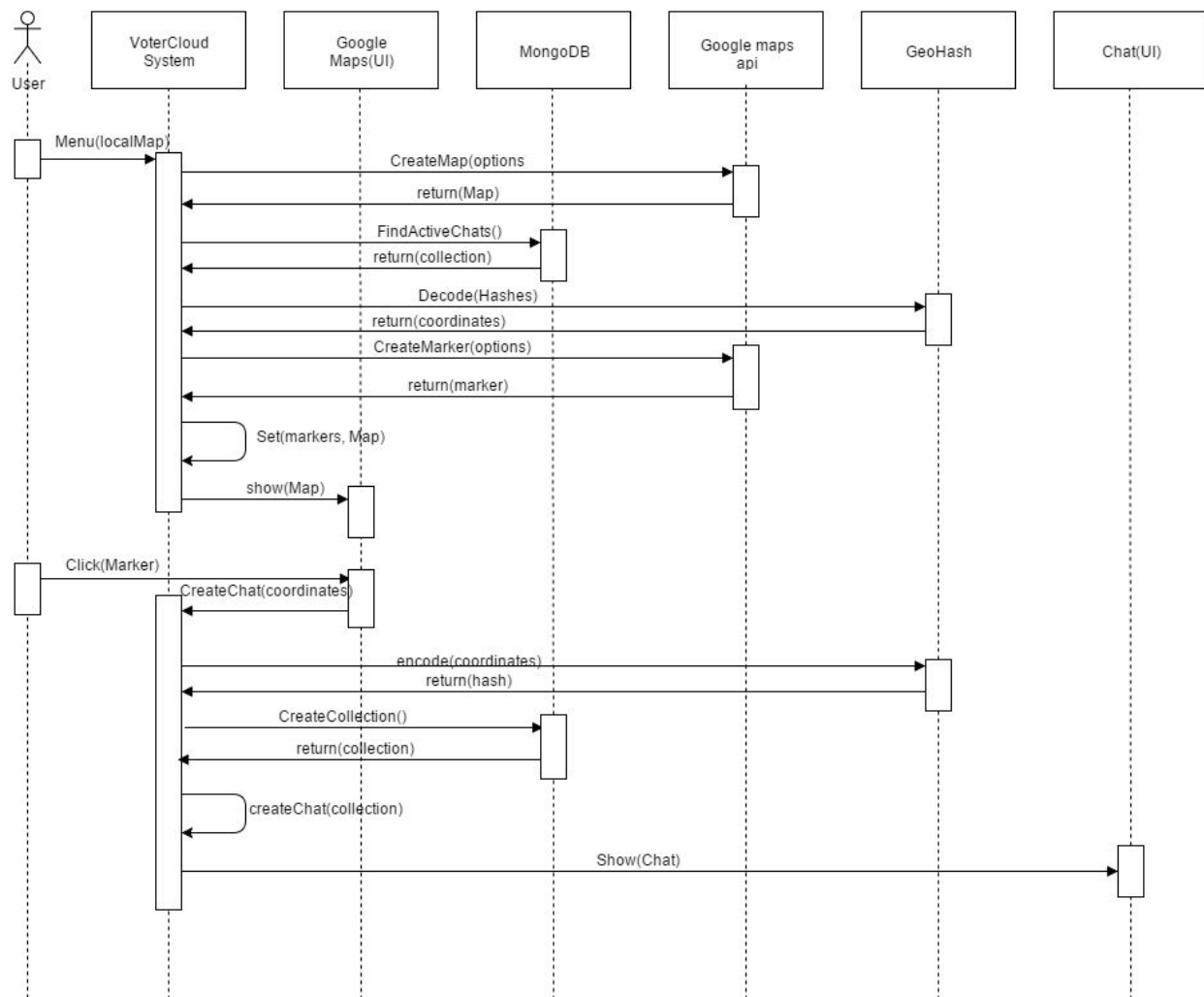


Figure 1.12 - The sequence diagram of the locality chat user story.

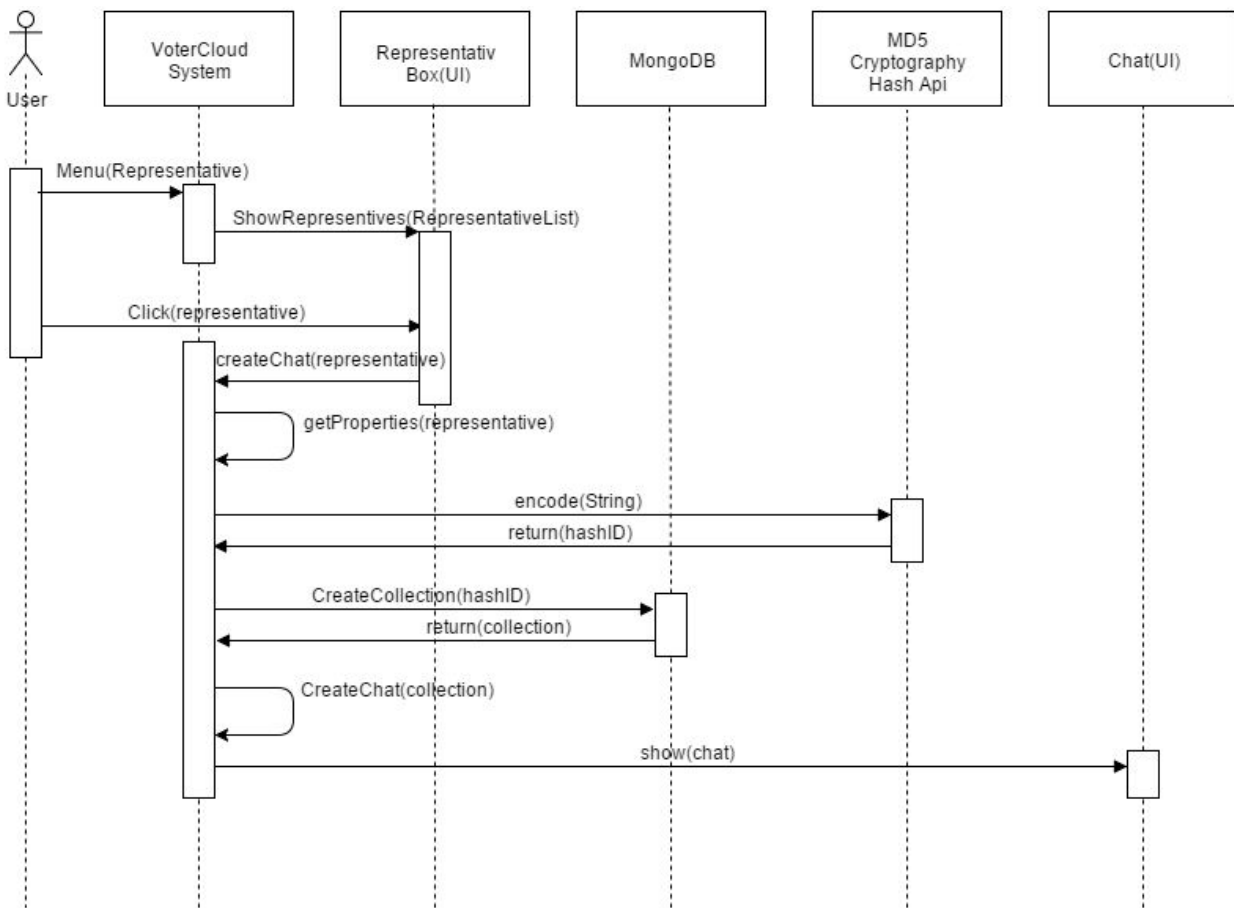


Figure 1.13 - The sequence diagram of the representative chat/page.

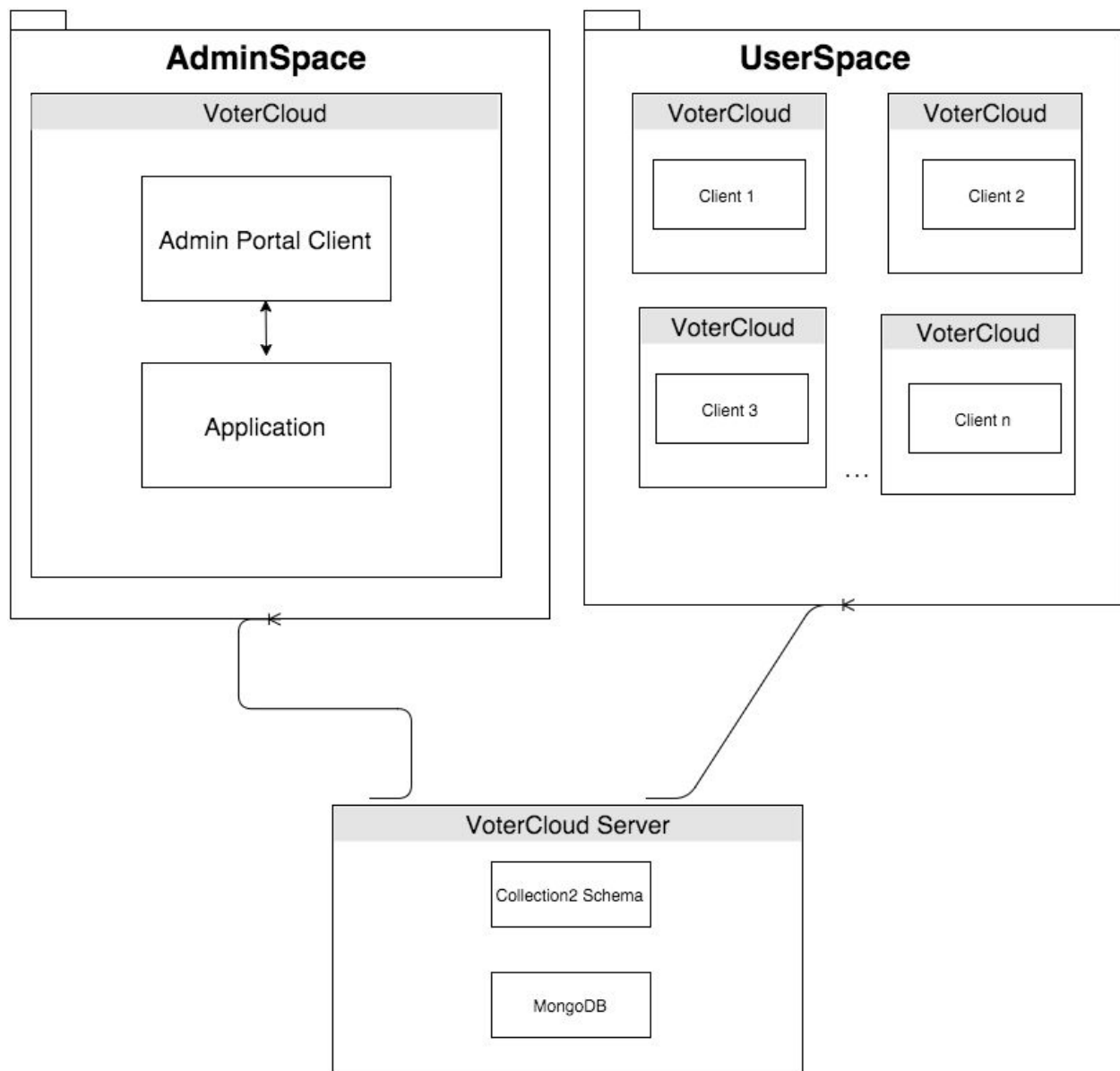


Figure 1.16 - Admin diagram, mapping between the subsystems of the portals.

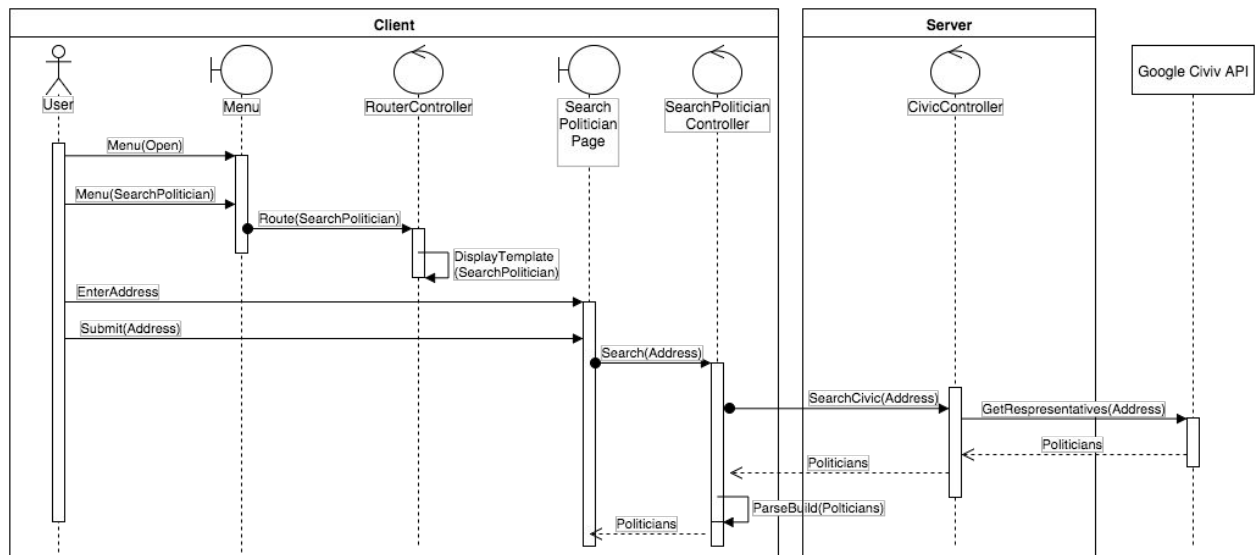


Figure 1.01 - Sequence diagram.

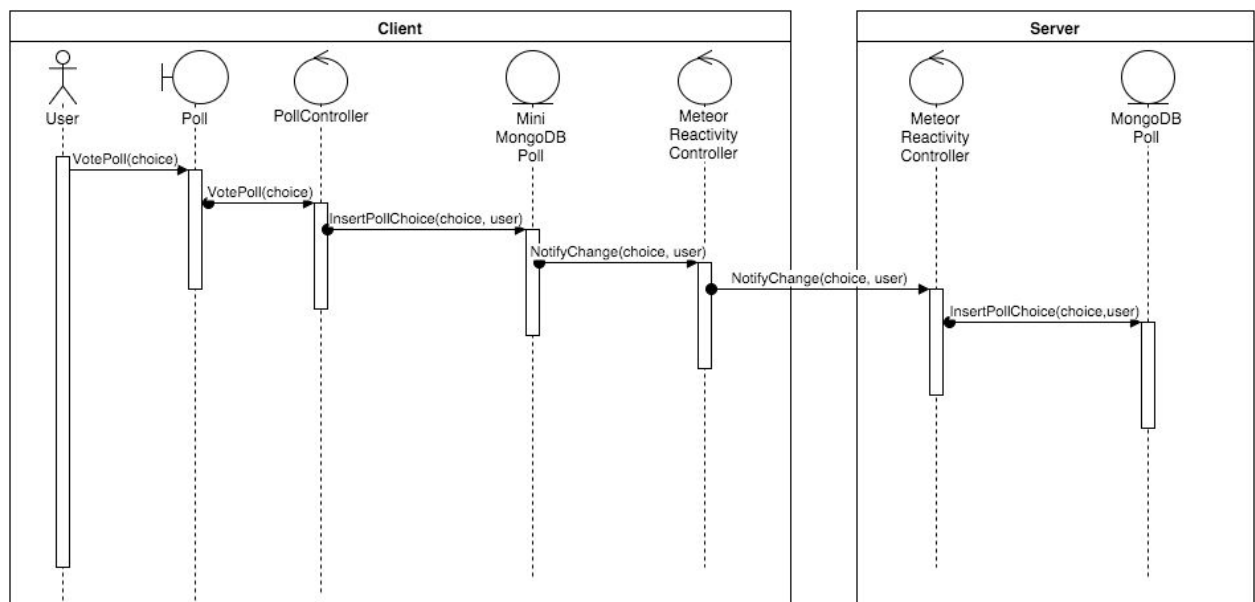


Figure 1.02 - Vote Poll

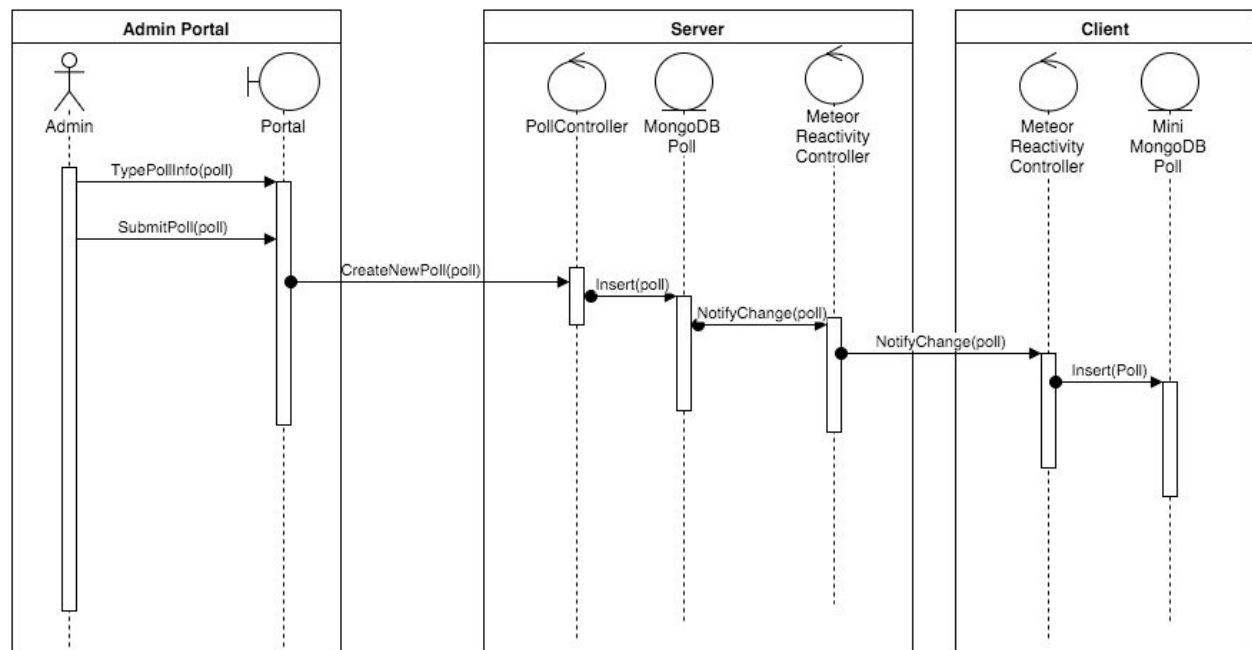


Figure 1.03 - Create Poll

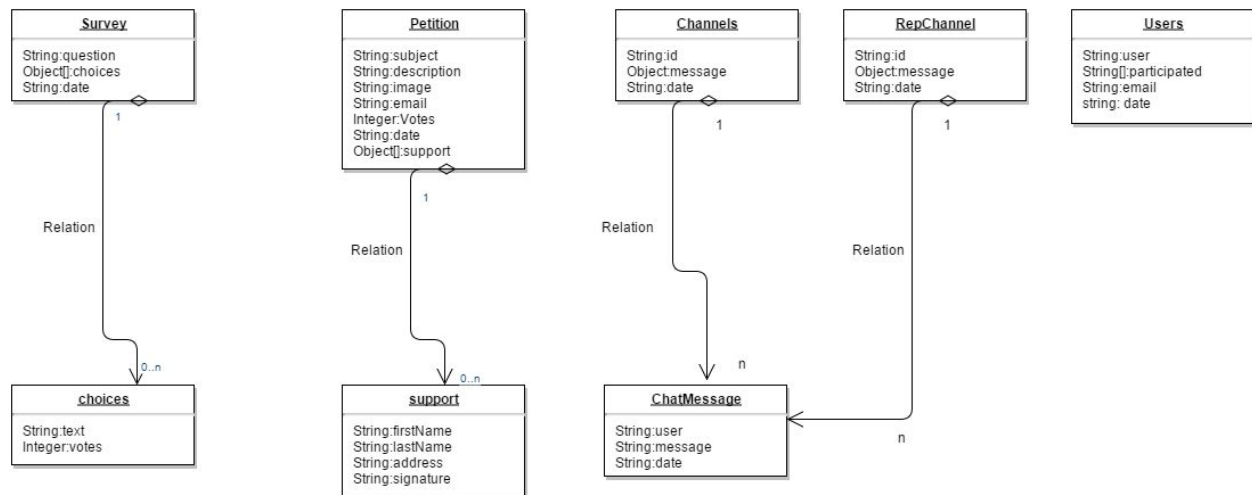


Figure 1.05: Database diagram Client and Server.

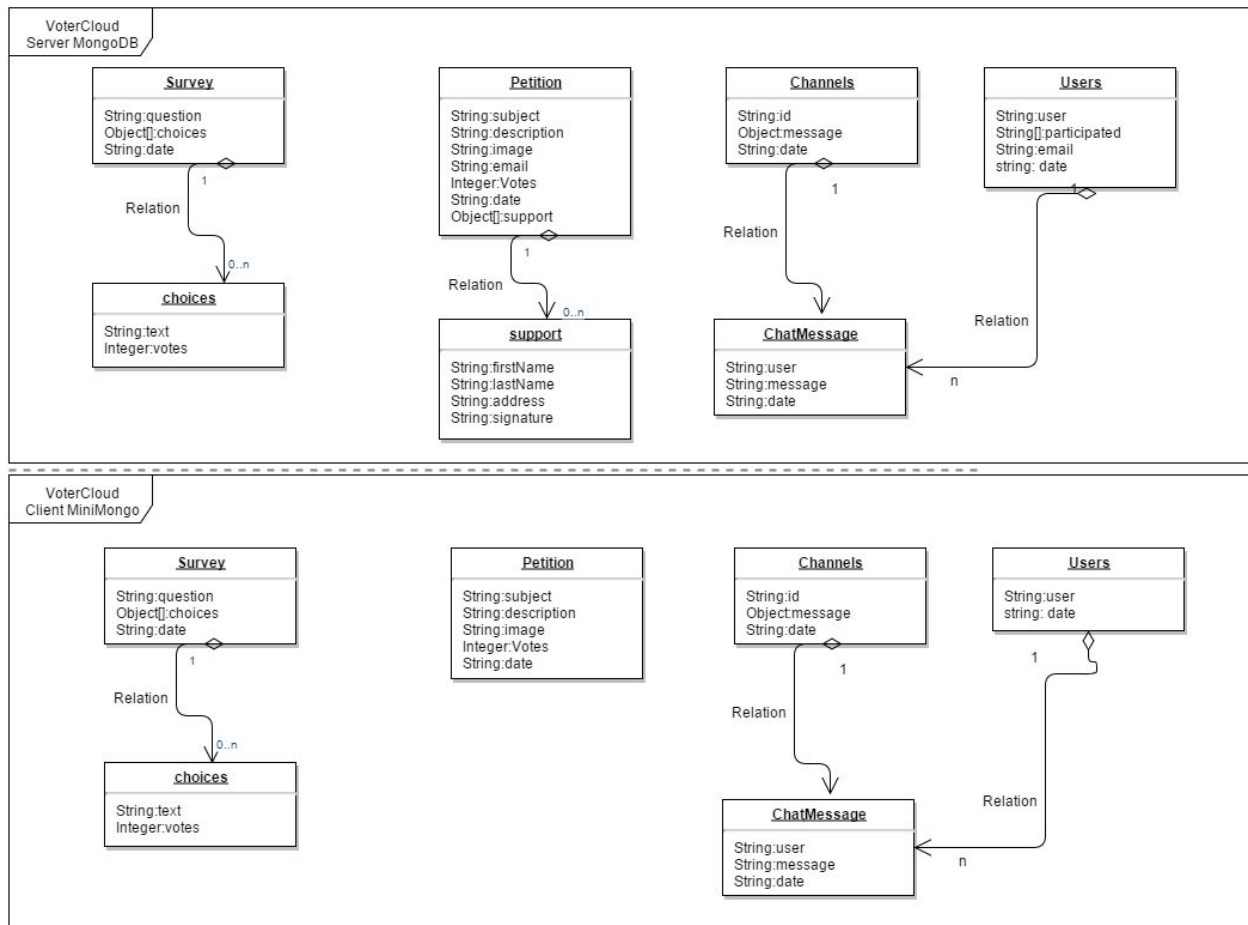


Figure 1.06: Database diagram Client and Server.

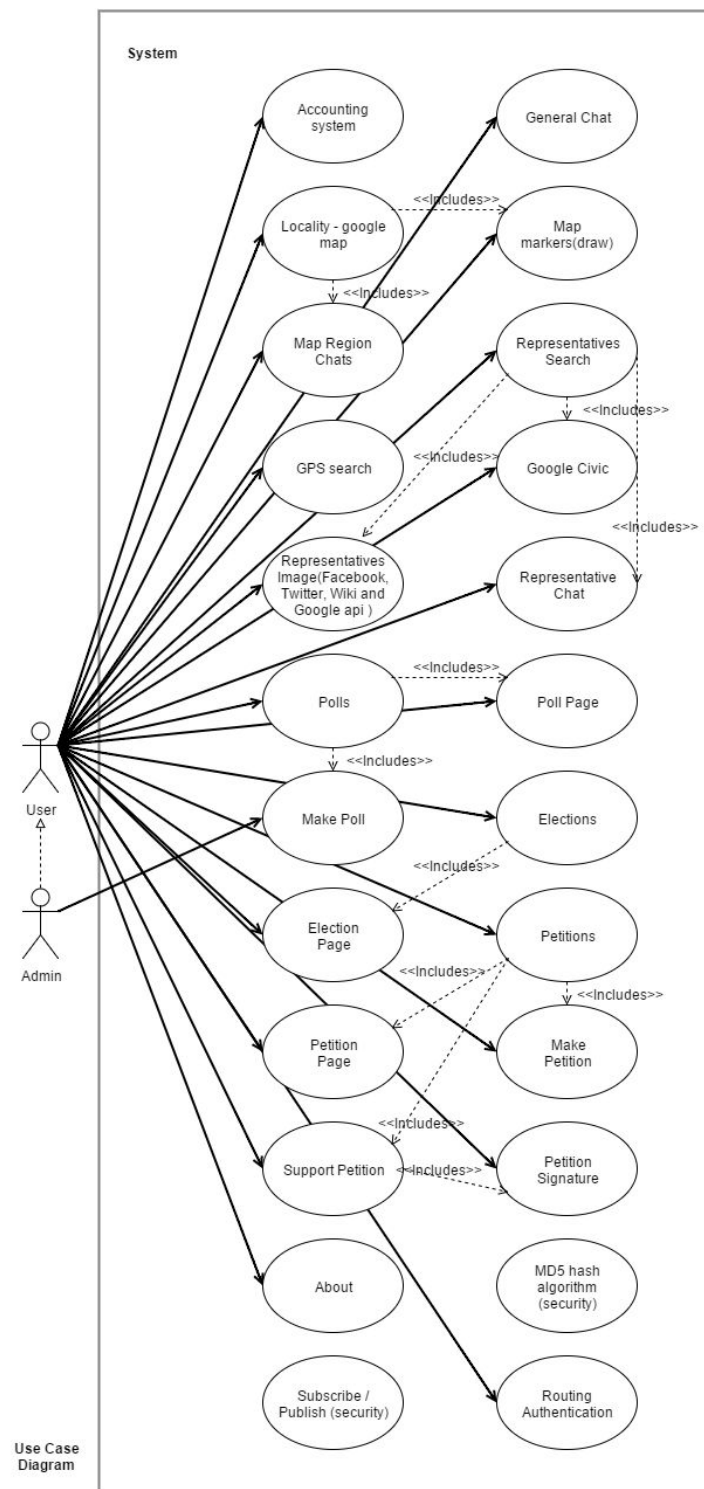
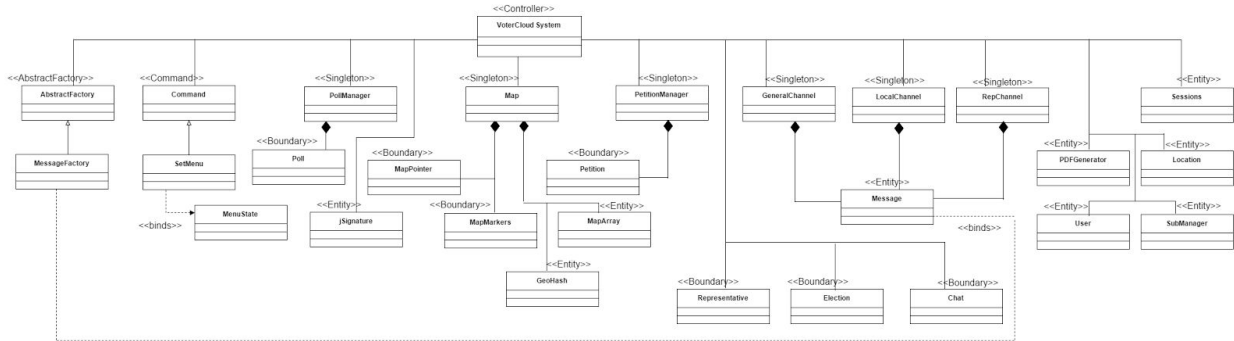
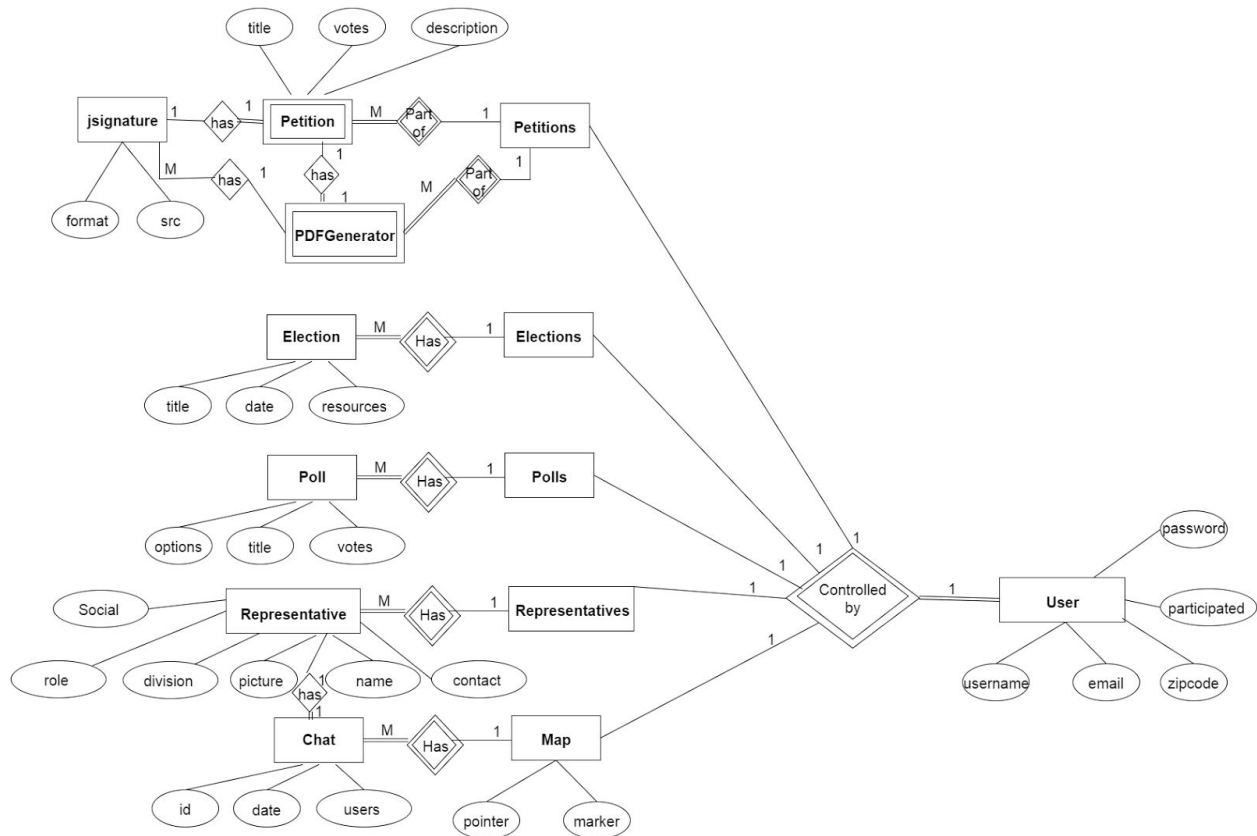


Figure 1.07: Use Case Diagram.**Figure 1.08: Class Diagram.****Figure 1.09: ER diagram.**

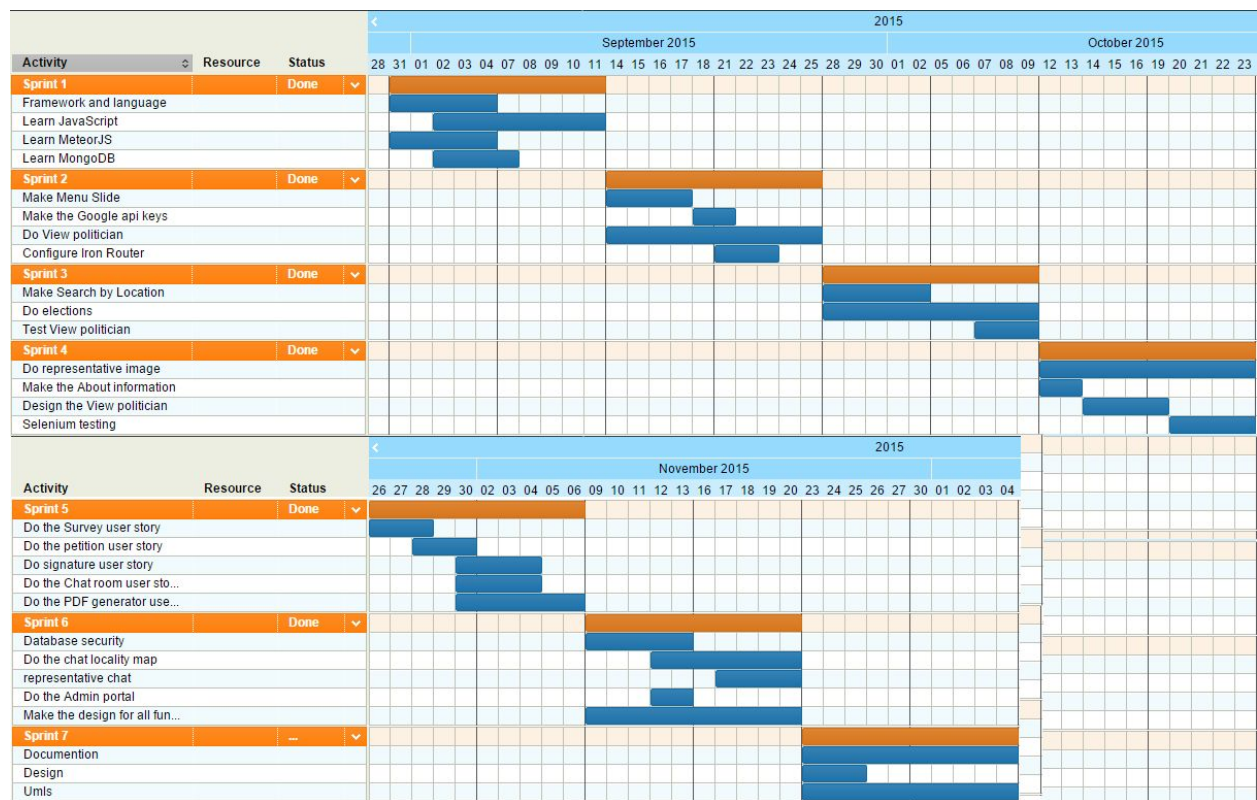
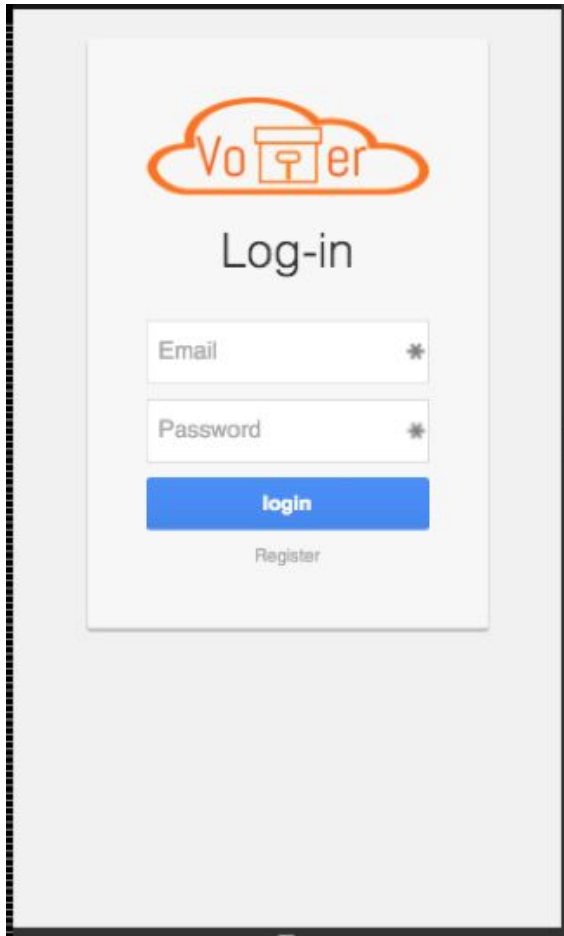


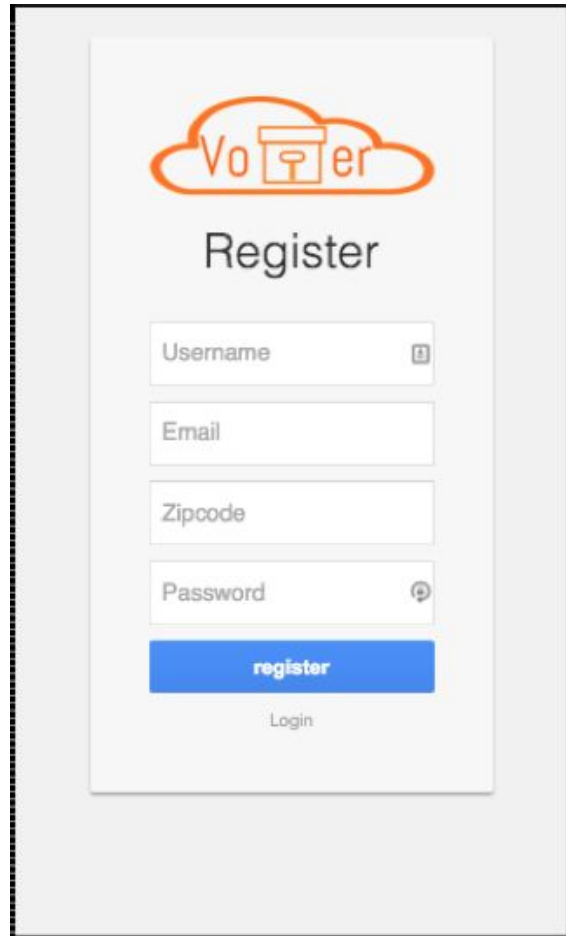
Figure 1.10: Gentt Chart.

Appendix B - User Interface Design



The Log-in page features the VoterCloud logo at the top, which consists of an orange cloud containing the word "Voter" and a ballot box icon. Below the logo is the heading "Log-in". The form includes two input fields: "Email" and "Password", each followed by an asterisk (*) indicating a required field. A blue "login" button is positioned below the password field, and a "Register" link is located at the bottom of the form.

Figure 1: The Log-In page



The Registration page features the same VoterCloud logo at the top. Below the logo is the heading "Register". The form includes four input fields: "Username" (with a user icon), "Email" (with an email icon), "Zipcode", and "Password" (with a password icon). A blue "register" button is positioned below the password field, and a "Login" link is located at the bottom of the form.

Figure 2: The Registration Page

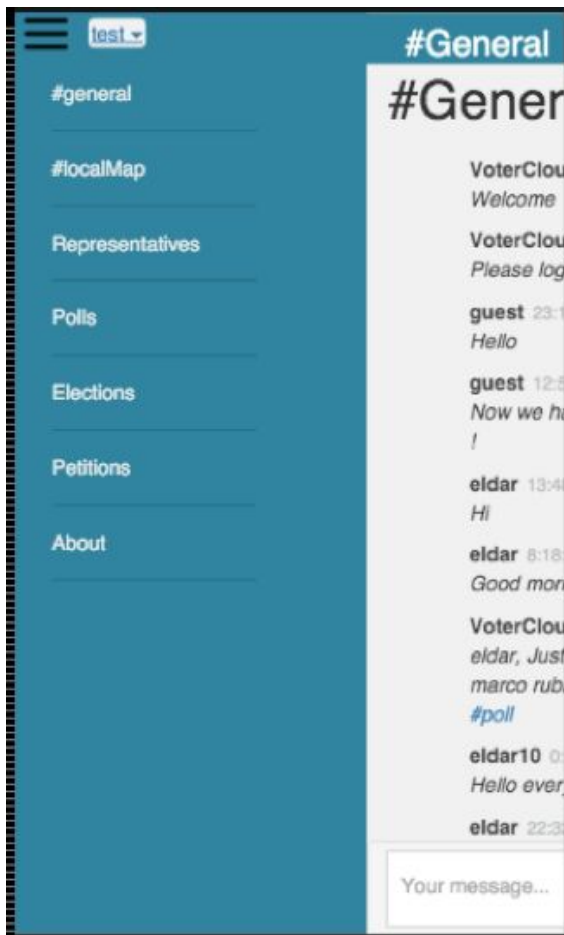


Figure 3: The slide-out menu

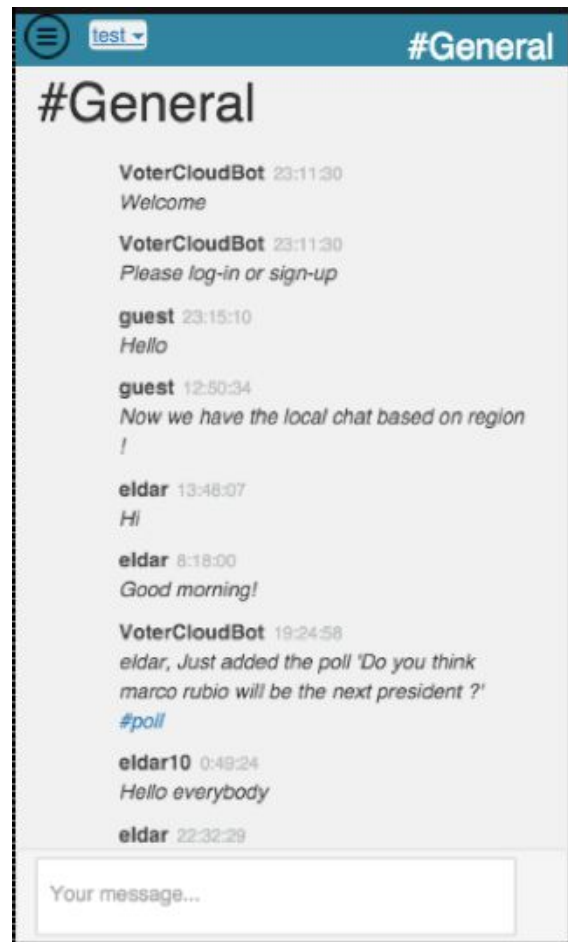


Figure 4: The home page. Displaying the central “general” chat.

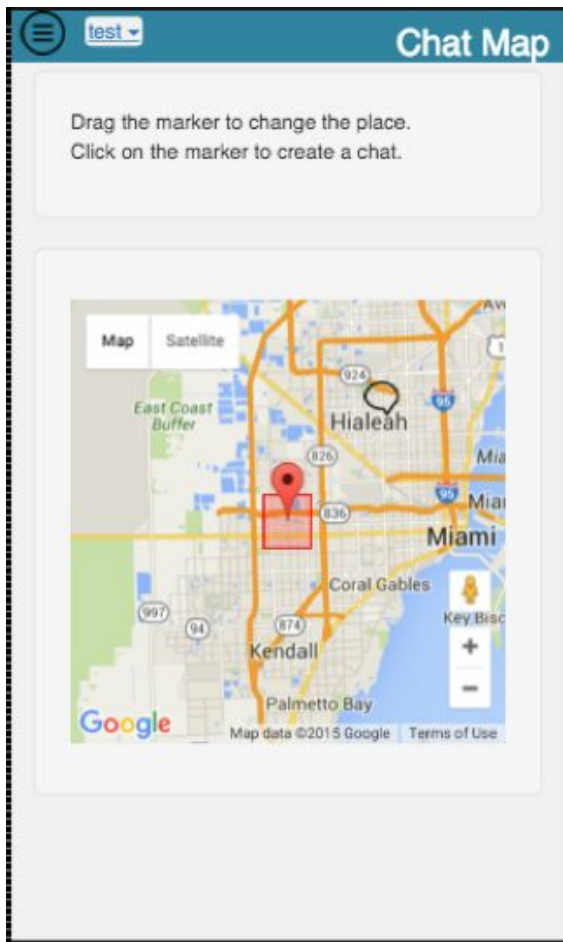


Figure 5: The Chat map that allows one to chat with others in the highlighted area.

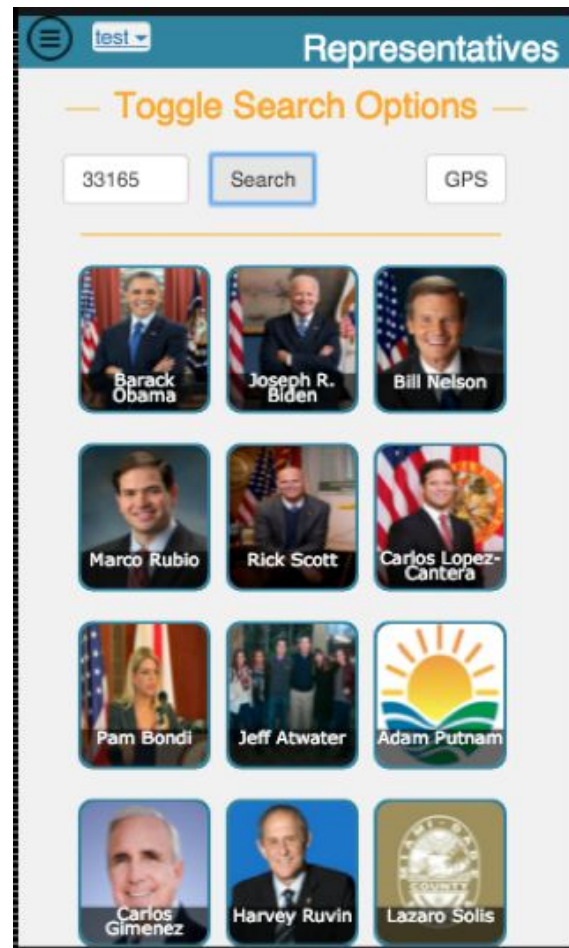


Figure 6: The Representatives search page. Once searched, a representative can be selected to see his/her individual page with chat functionality.

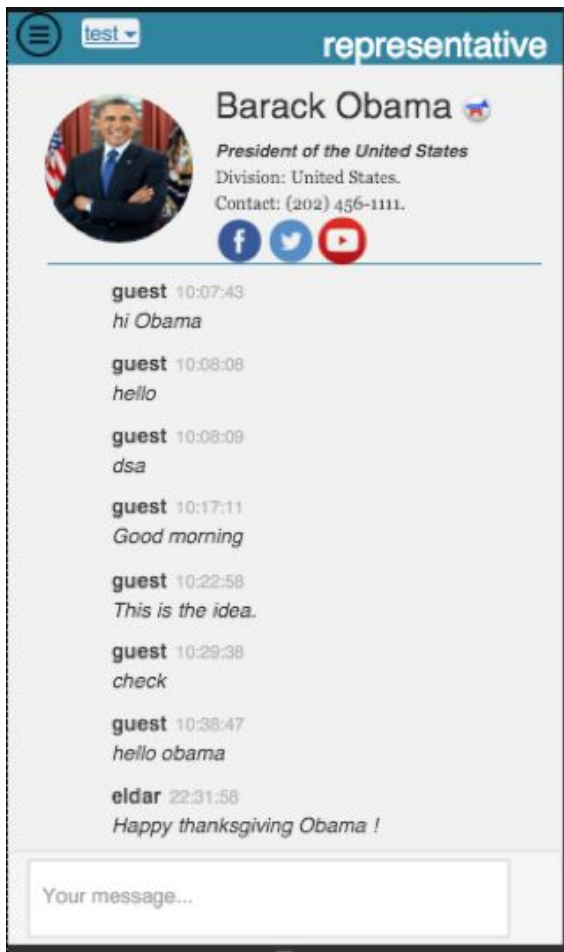


Figure 7: The Individual Representative page with chat functionality. The social media buttons take direct you to his/her respective profile.



Figure 8: The Polls page. It allows users to vote on polls as well as create them.

test Polls

Question

Answer #1

Answer #2

Answer #3

(optional) Answer #3

For more options +

Create Poll

Do you think marco rubio will be the next president ?

Link

Yes 3

Figure 9: The Create Poll page. Poll creation is a temporary functionality that will be reserved for the admin in future releases.

test Upcoming Elections

Virginia March Primary Election 2016-03-01

Resources

Absentee voting

Info

Registration Confirmation

Registration

Rules

Locations

Rhode Island Barrington School Election 2015-12-15

Figure 10: The Elections page. Shows all upcoming elections and some resources if available.

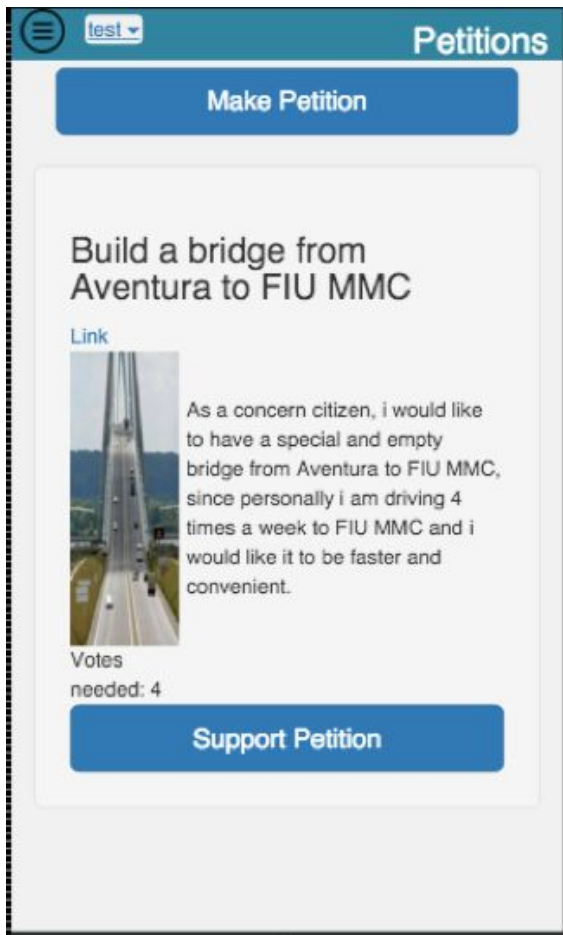


Figure 11: The Petitions Page. A user can view or select to create and support petitions created by others.

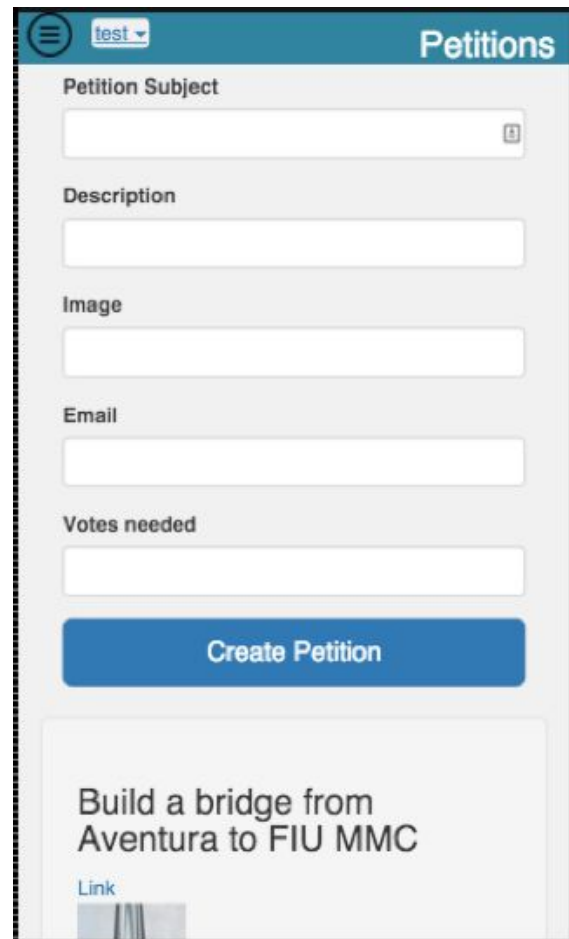
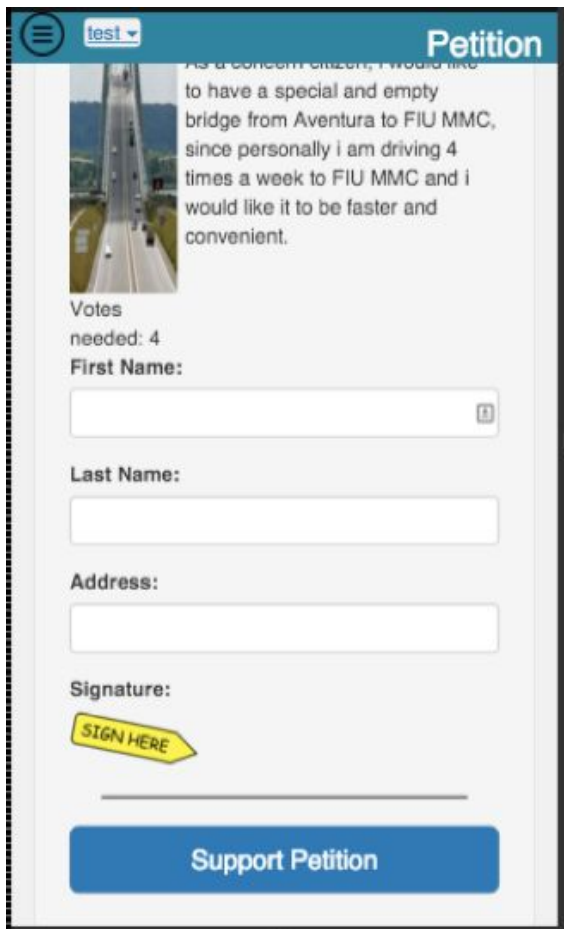


Figure 12: The Create Petition Page.



The screenshot shows a mobile application interface for supporting a petition. At the top, there is a blue header with a menu icon, a 'test' dropdown, and the word 'Petition'. Below the header, on the left, is a small image of a bridge. To the right of the image, the text reads: 'As a concerned citizen, I would like to have a special and empty bridge from Aventura to FIU MMC, since personally i am driving 4 times a week to FIU MMC and i would like it to be faster and convenient.' Below this text, it says 'Votes needed: 4'. There are four input fields: 'First Name:', 'Last Name:', 'Address:', and 'Signature:'. The 'Signature:' field has a yellow arrow pointing to it with the text 'SIGN HERE'. At the bottom, there is a blue button labeled 'Support Petition'.

Figure 13: The Support Petition page. A user fills out and signs the bottom digitally.

Build a bridge from Aventura to FIU MMC

As a concern citizen, i would like to have a special and empty bridge from Aventura to FIU MMC, since personally i am driving 4 times a week to FIU MMC and i would like it to be faster and convenient.

Votes Left: 0

• Eldar Feldbeine, Aventura, florida,

• Barack Obama, White House, DC,

• Donald Trump, Trump Tower, NC,

• Macro Rubio, Miami, Florida.,

Figure 14: The PDF of a Petition with all required Signatures.

Appendix C - Sprint Review Reports

Sprint 1 Report

Date: September 11, 2015

Attendees: Eldar Feldbeine, Raul Garay.

Discussed Topics:

This first sprint was focused on understanding the purpose, ideas, goals, and scope of VoterCloud with the mentor. We wanted to figure out the required system architecture as well as the languages and frameworks we initially know we will need to develop VoterCloud. These were all covered in the frameworks and languages tutorial, documentation, amazon web services user stories. However, towards the end of the Sprint, we decided to keep AWS as a our production host and use a simpler and faster hosting service for testing, staging, and development.

Sprint 2 Report

Date: September 25, 2015

Attendees: Eldar Feldbeine, Raul Garay.

Discussed Topics:

Once the hosting was settled, Eldar and Raul decided to build VoterCloud using the Meteor full stack framework. We performed several tutorials and showcased a sample menu for the application using Meteor. Since throughout the sprint we also studied the Google Civic API, we saw that it provided some valuable information that the application could take advantage of. So we presented several calls and returns of the Civic API in a page that is to set to become the search Representatives page.

Sprint 3 Report

Date: October 9, 2015

Attendees: Eldar Feldbeine, Raul Garay.

Discussed Topics:

After playing with the Google Civic API, we agreed to use it and further developed the Representatives page. We also showcased how the possible feature of using location based search by GPS would work. The GPS is planned to be used in the Representatives page for now. We also used the Civic API to build the Elections page which displays information about all upcoming elections in the near future.

Sprint 4 Report

Date: October 23, 2015

Attendees: Eldar Feldbeine, Raul Garay.

Discussed Topics:

The primary focus of this sprint was to finalize the last features of the Representatives page. We presented the politicians' images from their social media pages as well as links to their social media accounts. These were coming from the Google Civic API. Selenium testing was also presented since it was performed on all features implemented.

Sprint 5 Report

Date: November 6, 2015

Attendees: Eldar Feldbeine, Raul Garay.

Discussed Topics:

New pages were presented in the Sprint Review. The first is the Petitions page. Most functionality worked except the email to the Representative. However, an impressive feature presented was the signature capability for a user to sign the Petition he or she is supporting. The Polls page was also started and presented. A Poll could be manually created and can be voted on. We showed the count of how many votes have gone to each option in the respective poll. An admin portal was also attempted to be implemented, however, it became difficult with the Meteor framework to connect to a remote MongoDB Meteor database. A chat feature on the main home page was also presented. It uses the Slack application styling and similar "Channel" functionality.

Sprint 6 Report

Date: November 20, 2015

Attendees: Eldar Feldbeine, Raul Garay.

Discussed Topics:

A new admin portal was presented this sprint. It allowed the user to create, delete, and edit Polls. There was a bug in which when a poll is created through the admin portal, it breaks the Polls page, but deleting the poll "un-breaks" the Polls page. A new UI and color scheme was presented for the Representatives, Elections, and top menu bar as well. The Representatives Chat was finalized and was placed in an Individual Representatives page where if a representative is selected, his or her page will open with their information and chat. The new Localized Chat page was presented and showed a map with areas selected and a chat for each of these areas. It worked extremely well and showed how a feature developed late in the semester could be valuable.

Appendix D - Sprint Retrospective Reports

Sprint 1 Retrospective

Date: September 11, 2015

Attendees: Eldar Feldbeine, Raul Garay.

Discussed Topics:

In this sprint Eldar and Raul got to meet and know each other in terms of timing and skills, it's perhaps the basis of our common success on this project. The Product owner and mentor guide as with the concept and target system (meteorjs). In the beginning we had a debate how and what we gonna implement. Since this project involves deep and complicated implementations and functions. The specific actions Eldar and Raul did during this sprint is learning the new language and Meteor methods. Therefore the actions agreed on are the process of learning and the path to succeed in this project. The main challenges we faced are that this framework is still relatively new in terms of expression and community. and having this small framework community effected in a way the challenges in learning the language.

Sprint 2 Retrospective

Date: September 25, 2015

Attendees: Eldar Feldbeine, Raul Garay.

Discussed Topics:

In this sprint Eldar and Raul got to review our process in terms of what we learned and where we go. during this sprint we agreed on the main user stories we will get to implement during this semester and what that involves. The sessions went well especially in the social dynamic with the mentor and product owner , so that our communication went well to understand the basic process and functionalities we will implement. We agreed upon the user stories we will implemented and the plan how we will achieve that (by first start the base functionalities so that we can learn and develop using MeteorJS). there not yet any issues to the next sprint.

Sprint 3 Retrospective

Date: October 9, 2015

Attendees: Eldar Feldbeine, Raul Garay.

Discussed Topics:

In this sprint Eldar and Raul got to do a real and deep functionalities using the process done during the last two sprints. we implemented during this sprint the elections and search politician user stories. we agreed on what Apis will be used and how we will do it. The api's agreed in this sprint are the google civic api. we succeed to fetch and create google key to the application. using the key we were able to parse the data from the api and serve it as data to the user using

user friendly view. In this sprint everything went relatively smoothly and successfully, we agreed on how we gonna implement that and achieve that using our new set skills. the challenges for the next sprint is still knowledge gap with the new language.

Sprint 4 Retrospective

Date: October 23, 2015

Attendees: Eldar Feldbeine, Raul Garay.

Discussed Topics:

In this sprint Eldar and Raul got to do the search representative by image and menu. This sprint was relatively hard in terms of complexity and bugs. Here we discover the real meaning of small framework community, given that the meteorjs small community it was hard and difficult to resolve and fix bugs in our system. Therefore this sprint was dedicated fixing the bugs and apis, packages conflict we experience. This sprint we indeed implemented the search by image representative, which involves different apis such as facebook, wikipedia and twitter apis. We agreed on the apis used and how the efficiency and view should be. Therefore, this sprint was crucial for the continue of the next sprints in terms of functionalities and understanding.

Sprint 5 Retrospective

Date: November 6, 2015

Attendees: Eldar Feldbeine, Raul Garay.

Discussed Topics:

In this sprint Eldar and Raul did the petition , petition pdf, polles, and individual pages. This sprint was the most productive sprint in terms of functionalities implemented and what achieved. this sprint everything went well in terms of what was achieved and implemented. The things in this spirit that were very challenging and didn't well as expected is the PDF generator and email sending. We found out that the anti virus prohibit and restrict you to send a file through email(attached file) , so except this small detail everything went all in this sprint. We agreed on how it will be organized and viewed, the actions here involves a lot of testing and validation. there not issues that are carried to the next sprint (except the pdf generator).

Sprint 6 Retrospective

Date: November 20, 2015

Attendees: Eldar Feldbeine, Raul Garay.

Discussed Topics:

This sprint was the last sprint in terms of functionalities and real team cooperation and organization about the project. During this sprint we implemented the map chat and chat for each representative. In addition during this sprint we lay down the main design of the application. Everything went well during this sprint we communicated a lot and cooperated a lot in terms of

final touches and how it should look like. we agreed upon the design and how it should look and function (the application). There no more issues except some little css design issues and bugs.

REFERENCES

No reference, since MeteorJS is very small community of developers, we had no reference of help from outside (this very fact was one of our challenges, and this is why we are so proud on our code and originality).