ChannelX

Project Summary Report

Istanbul Technical University
Computer Engineering Department

Team Members

Birkan Denizer
Kağan Özgün
Berkay Giriş
Muhammed Alperen Özkan
Berk Dehrioğlu

Project Advisors

Dr. Ayşe Tosun (Instructor)
Müge Erel Özçevik (Teaching Assistant)

Table of Contents

1. Project Charter

1.1. Description

Today, many of the available communication means require users to be identified. However, they neither provide a temporary connection between users nor preserve the personal information securely. This project aims to help users communicate with other users without exposing their personal contact information. The proposed communication channels will be generated by users of this system and the channels will be available for chat for a pre-specified period of time. With the help of this proposed system, users are able to protect their identities, and they don't need to be available for chat anytime.

The team *HERMES* from the Computer Engineering Department at Istanbul Technical University, consisting of senior undergraduate students have built a web application. In this report, we will explain the details of analysis, requirements, design and implementation stages of our development life cycle. As the deliverables of the Software Engineering course, all students enrolled into the course need to choose a viable project to be analyzed and implemented. HERMES team picked Channel X from ICSE SCORE Competition Website [1]. One of the Teaching Assistants (TA) of the course was acting as the customer of the project. All the customer meetings that will be mentioned in this report took place between the team and the TA.

The project will be considered a success if the team delivers a functional, reliable and documented web application for the customers in which anonymous transient shared communication channels are provided. The project plan is described in the project's Gantt chart.

1.2. Project Scope

The proposed web application allows users to communicate with each other using transient channels which have anonymous members. Three types of channel privacy options are presented: public, private and hidden. The channels also have the options for e-mail and SMS notification for all the members depending on their communication preferences. The channel owner has the right to ban, unban and delete a channel member. The system presents users a calendar feature which demonstrates owned or joined channel sessions in a user interface. System also gives the users the options of specifying a favorite channel and favorite messages. If a user marks a message or a channel as his/her favorite, the user can see them on their own profile page.

The project was implemented with all its functionalities between September 2017 and January 2018, and it was delivered with a demo during the course. Project deliverables in accordance with the course plan are Project Plan, Requirement Specification, Design Document, Testing Report Minimum, and Viable Product Presentation and Report.

1.3. Team Members and Functions

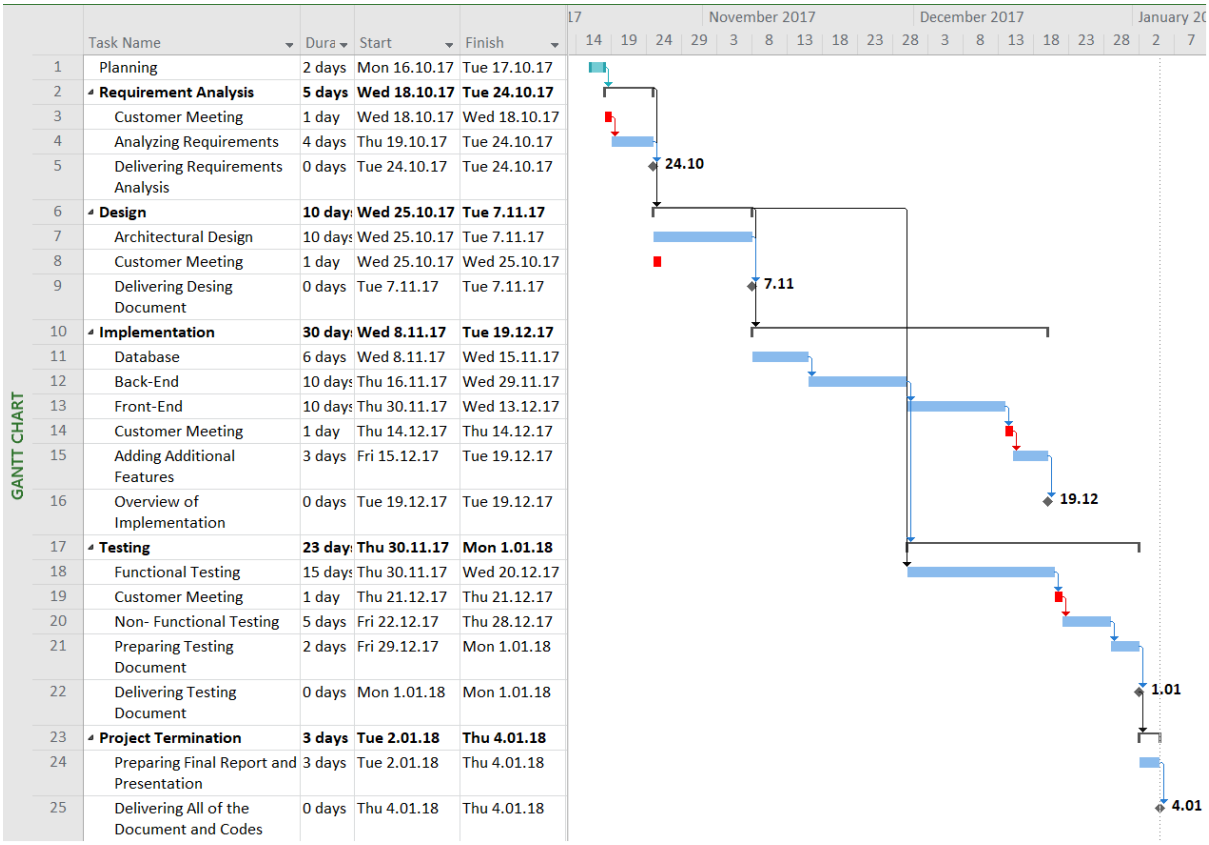
We formed the team on a voluntary basis. The team roles were formed in an agile way. Although each team member has predefined roles and responsibilities, all members worked collaboratively at every stage of the development. The responsibilities and roles are set as follows: Birkan is the project manager and he is responsible for planning and managing the project. He also implemented external e-mail service into the system. Berkay is the lead developer and he is mainly responsible for implementing core functionalities such as in-app messaging, channel creation and

asynchronous SMS and e-mail handling. Berk is the designer and he is responsible for designing user interface, database model and related interactions that are happening. He also took part in integration testing step of the project. Alperen is the back-end developer and his contribution to the project is managing the interaction that is happening between database and server. Kağan is the tester of this project and he is the architect and manager of the all testing related functionalities. He also took part in development by implementing external SMS service into the system.

1.4. Timeline

We constructed our project Gantt chart including six main tasks: planning, requirement analysis, design, implementation, testing and project termination. There are milestones after each of the main tasks. During the project development period, four meetings were made with the customer. After every weekly lab session of the course, it was possible to meet with the TA. In total, the team made four face-to-face meetings with the TA to present the work-in-progress product, deliverables and discuss the issues faced.

The detailed version of the Gantt chart and the Microsoft Project file is accessible through https://goo.gl/6Y8wo1. Although the chart follows a waterfall development which is in accordance with the course deadlines, the team worked with weekly sprints and retrospective meetings in every two days.



	Task Name	Dura	Start	Finish
1	Planning	2 days	Mon 16.10.17	Tue 17.10.17
2	**Requirement Analysis**	**5 days**	**Wed 18.10.17**	**Tue 24.10.17**
3	Customer Meeting	1 day	Wed 18.10.17	Wed 18.10.17
4	Analyzing Requirements	4 days	Thu 19.10.17	Tue 24.10.17
5	Delivering Requirements Analysis	0 days	Tue 24.10.17	Tue 24.10.17
6	**Design**	**10 days**	**Wed 25.10.17**	**Tue 7.11.17**
7	Architectural Design	10 days	Wed 25.10.17	Tue 7.11.17
8	Customer Meeting	1 day	Wed 25.10.17	Wed 25.10.17
9	Delivering Desing Document	0 days	Tue 7.11.17	Tue 7.11.17
10	**Implementation**	**30 days**	**Wed 8.11.17**	**Tue 19.12.17**
11	Database	6 days	Wed 8.11.17	Wed 15.11.17
12	Back-End	10 days	Thu 16.11.17	Wed 29.11.17
13	Front-End	10 days	Thu 30.11.17	Wed 13.12.17
14	Customer Meeting	1 day	Thu 14.12.17	Thu 14.12.17
15	Adding Additional Features	3 days	Fri 15.12.17	Tue 19.12.17
16	Overview of Implementation	0 days	Tue 19.12.17	Tue 19.12.17
17	**Testing**	**23 days**	**Thu 30.11.17**	**Mon 1.01.18**
18	Functional Testing	15 days	Thu 30.11.17	Wed 20.12.17
19	Customer Meeting	1 day	Thu 21.12.17	Thu 21.12.17
20	Non- Functional Testing	5 days	Fri 22.12.17	Thu 28.12.17
21	Preparing Testing Document	2 days	Fri 29.12.17	Mon 1.01.18
22	Delivering Testing Document	0 days	Mon 1.01.18	Mon 1.01.18
23	**Project Termination**	**3 days**	**Tue 2.01.18**	**Thu 4.01.18**
24	Preparing Final Report and Presentation	3 days	Tue 2.01.18	Thu 4.01.18
25	Delivering All of the Document and Codes	0 days	Thu 4.01.18	Thu 4.01.18

Figure 1 - Gantt chart of the development cycle

1.5. Project Risks

Risks that could be encountered during the project were identified during the planning phase and mitigation plans were defined regarding each of the risk (Table 1).

Risk	**Probability**	**Impact**	**Mitigation**
Losing a team member	Low	Medium	Every team members' continuation to the project will be checked weekly as it is required by the course rules.
Deviations from the plan	Low	Severe	The plan is in accordance to the course schedule and it is strictly followed throughout the semester. Weekly retrospective sessions will be followed to keep track of the progress.
Disagreement with SMS or email provider	Low	Medium	Alternative service provider could be selected considering the cost and time.
Data Security Risk	Medium	Severe	Reliable third-party services with respect to security protocols such as TLS and SSL will be selected.

Table 1 - List of possible risk and proposed mitigations

2. Overview of Requirements

The requirements of this projects are divided into two subcategories. In the first part, usage scenarios will be explained with user types and use cases with diagrams. In the second part, early system models will be clarified with a conceptual model and flow diagrams of the functions in the application.

The main features of the system are depicted in the form of use case diagrams, system models, and data flow diagrams. Some features are specifically picked for the representation of use cases, whereas others are visualized through data flow diagrams. This way we aim to present as many use cases as possible in this report.

2.1. Functional Requirements

All functional requirements of the project are specified and analyzed under four headings: User Types, Conceptual Model, Use Case Diagrams and Use Cases.

2.1.1. User Types

The application has three types of users: Registered User, Channel Owner, and Channel Member.

2.1.1.1. Registered User

A registered user is the most basic type of the user in the software. Users of this title can search for channels and users, create channels and join channels.

2.1.1.2. Channel Owner

A registered user can become a channel owner by creating a channel. The channel owner extends abilities of the registered user type. The users of this title are able to send and receive messages, favorite channels and messages, set properties for the channel (privacy, time schedule etc.), invite, and kick or ban users from the channel. The channel owner also can delete a channel if such a channel is no longer needed.

2.1.1.3. Channel Member

A registered user can become a channel member by joining to a public, private or hidden channel. Such a user can participate in conversations of private or hidden channels after a channel owner invites that user to the respective channel. The channel member extends the abilities of the registered user. The users of this title are able to send and receive messages, favorite channels, and messages.

2.1.2. Conceptual Model

Figure 2 shows the conceptual model of the application with the user roles, the actions that can be done by each of the users, and the communication with the system's main components (website, database, external service providers).

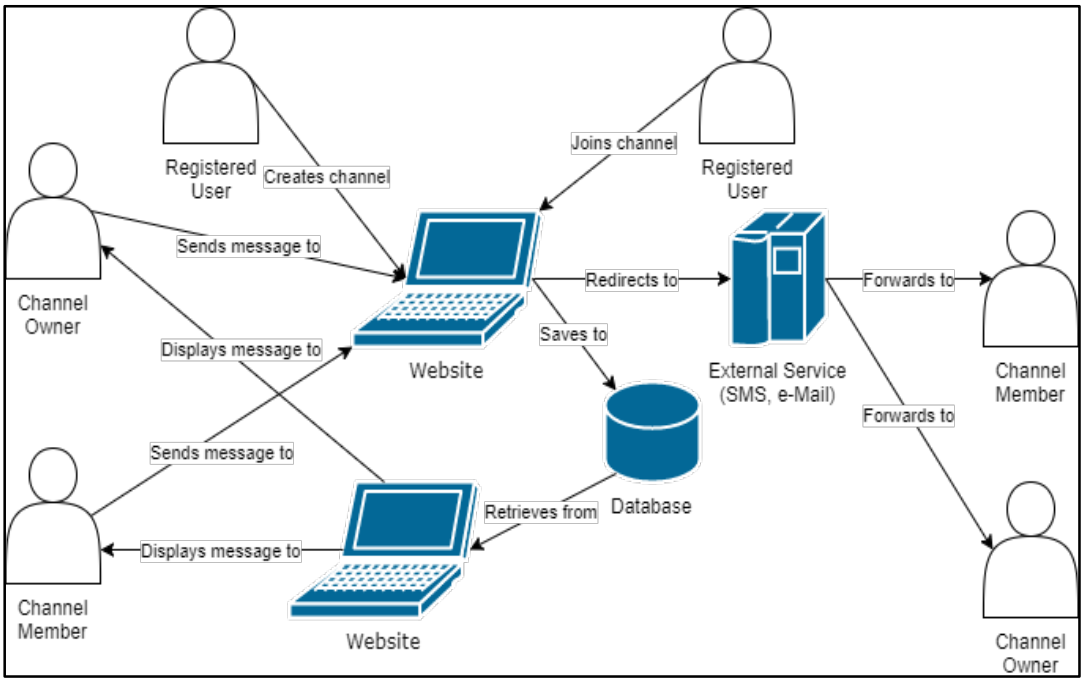


Figure 2 - Conceptual Model of the system

2.1.3. Use Case Diagrams

Overview of the system with the main functionalities are presented in Figure 3. Each of these features (nodes in Figure 3) have more detailed use case diagrams, which can be accessed in https://goo.gl/am9o5A.

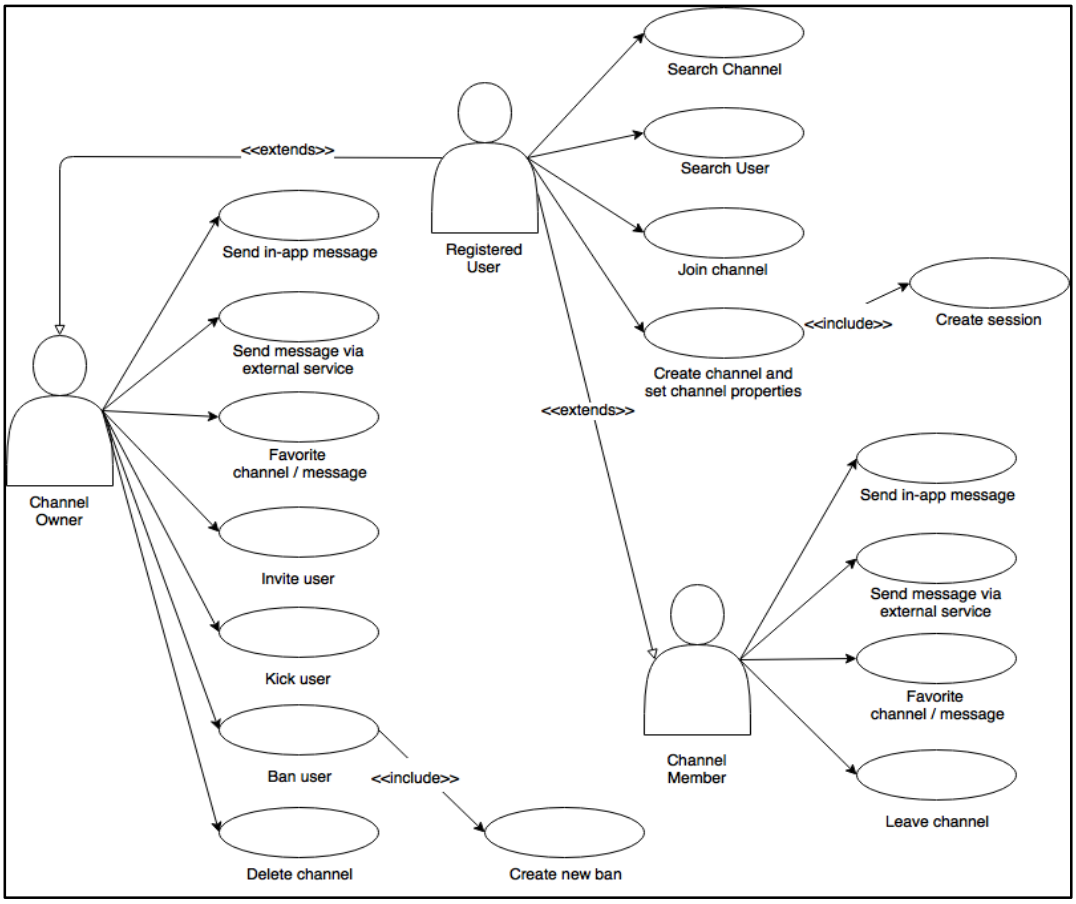


Figure 3 - Available operations for different user types

2.1.4. Use Cases

In this section, detailed use cases will be explained. Due to the page limit of the report, only four core features in these use cases will be presented. More use cases can be reached via https://goo.gl/LiHBUJ.

Use Case Number	1	3.1
Title	Search Channel / User	Create Channel
Purpose	Finding users or channels	Creating new channels
Overview	The use case starts when the search bar is used by entering a query. Search results are listed for the user.	The use case starts when the create button is clicked. A new channel is created and user is asked to enter its properties.
Actor	Registered user	Registered user
Preconditions	- Web application and server should be able to connect to each other for the query to work.	- Web application and server should be able to connect to each other for query to work. - Channel's properties must allow

		users to join.
Postconditions	- The result of the search is listed. - Search action will be saved into a log file.	- Channel is created. - Create action will be saved into a log file. - User is now an owner of the new channel. - User is brought up to set properties page.
Triggers	The user presses enter in the search bar.	The user presses the create button.
Normal flow of events	1) User may make a search from any place in the application with using the search bar. 2) Searched text is taken 3) Search choice (user/channel) is taken 4) Result is retrieved from the database and listed to the user.	1) User clicks the create button. 2) User gets brought up to set properties page.
Alternative flow of events	1) The system displays an error message if the search did not end successfully.	1) The system displays an error message if creation cannot be done.

Table 2 - Search Channel/User and Create Channel Use Cases

Use Case Number	6	10
Title	Invite User	Send Message via External Service
Purpose	Inviting new members to the channel	Sending message via e-mail or SMS
Overview	It starts when user clicks invite button on the channel. System provides search service to user. Inviter selects a user. System sends invitation to user.	The use case starts when the user wants to send message via SMS or e-mail.
Actor	Channel owner	Registered user (Member of an SMS or e-mail channel)
Preconditions	- Web application and server should be able to connect. - There should be existing channel.	- Sender's web application, the external service (SMS or e-mail) and the server should be able to connect to each other for query to work. - Recipient's e-mail or cell phone number should be present on the system.
Postconditions	- Invitation should be sent to user.	- A successful message request will be displayed to user and will be inserted into

		recipient's database file. - Sender's information which is registered on application will be added to message.
Triggers	Channel owner presses invite button.	The user sends message via external messaging service
Normal flow of events	1) Program takes username 2) User is searched and found on database. 3) Invitation is sent to user.	1) The user sends message via SMS or e-mail 2) System routes incoming message request to external service. 3) External service processes the request and sends message to targeted user and system. 4) The system inserts successful message into database if message is successfully sent to user.
Alternative flow of events	1) System cannot connect to database and displays an error message. 2) User cannot be found in the database and error message is shown to user.	1) System cannot connect to external service and displays error an error message. 2) External service cannot process the request and an error response is sent back to system.

Table 3 - Invite User and Send Message via External Service Use Cases

2.2. Non-functional requirements

The most basic data such as the username and password shall be protected using various hashing algorithms. Additionally, the data sent to third-party services (e.g. messages) shall be protected by encryption algorithms such as TLS or SSL and hashing algorithms such as HMAC-SHA1. User interface of the web application shall be user-friendly and will contain intuitive and simple controls.

3. Overview of Design

The aim of this section is to provide a description of project's inner workings, providing insight into the project structure and its components. In short, the section is meant to grant the reader with a solid understanding of the low-level organization of the system.

3.1. Architectural Design

Since Model-View-Controller architecture allows designing loosely coupled systems and provides separation of concerns in web development, it was chosen as the application's high-level architecture model [2]. The application interacts with two external message services. These services are identified in Section 4.

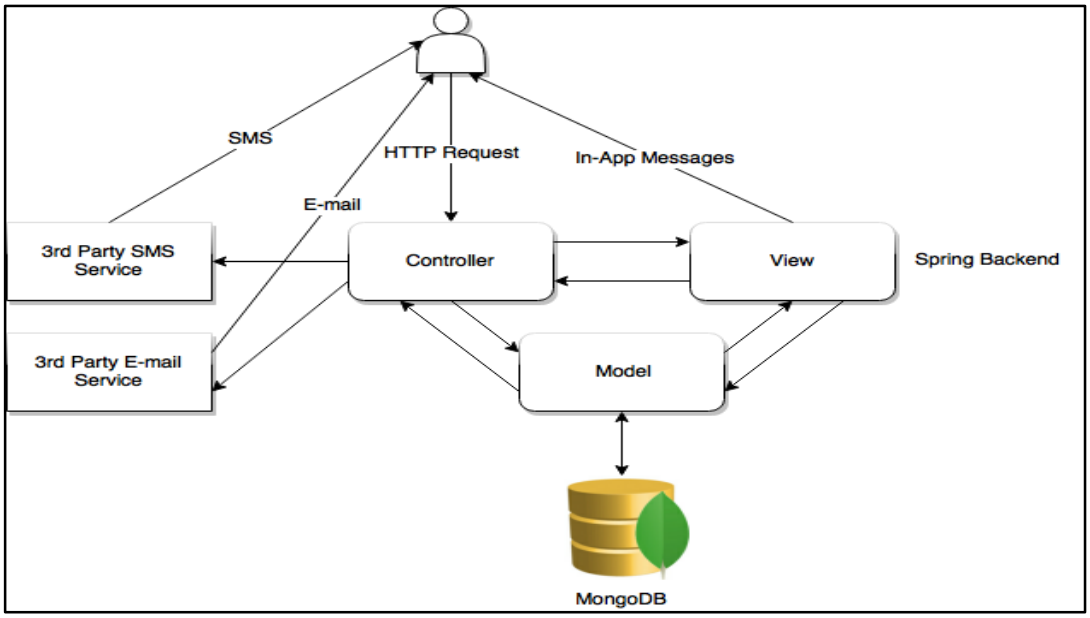


Figure 4 - High Level Architecture of the system

3.1.1. Flow Diagrams

In this section, the flow diagrams related to the operations between the three layers of MVC model are presented. As the main utility of this system is either creating or joining to a channel, channel creation (Figure 6) and channel search (Figure 1) plays a crucial role in this design. If a channel is private or hidden, channel owner needs to invite users (Figure 7) to such channels so that conversation (Figure 8) can happen. Due to the page limit of the report, only four core features in flow diagrams are presented. Additional flow diagrams can be reached via https://goo.gl/KRUoSX.

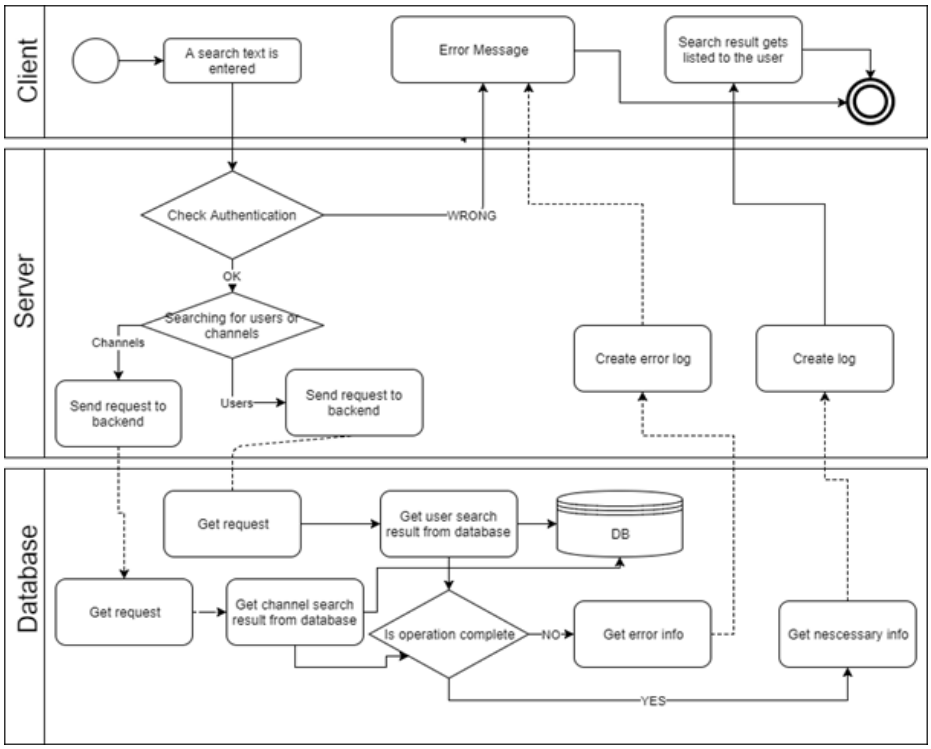


Figure 5 - Search Channel / User

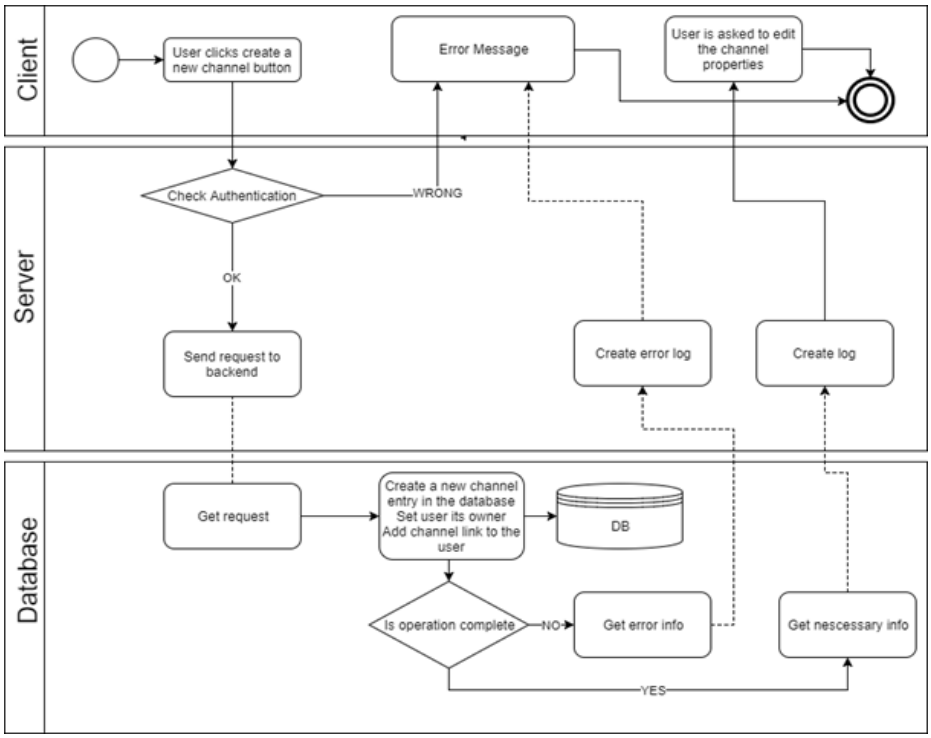


Figure 6 - Create Channel

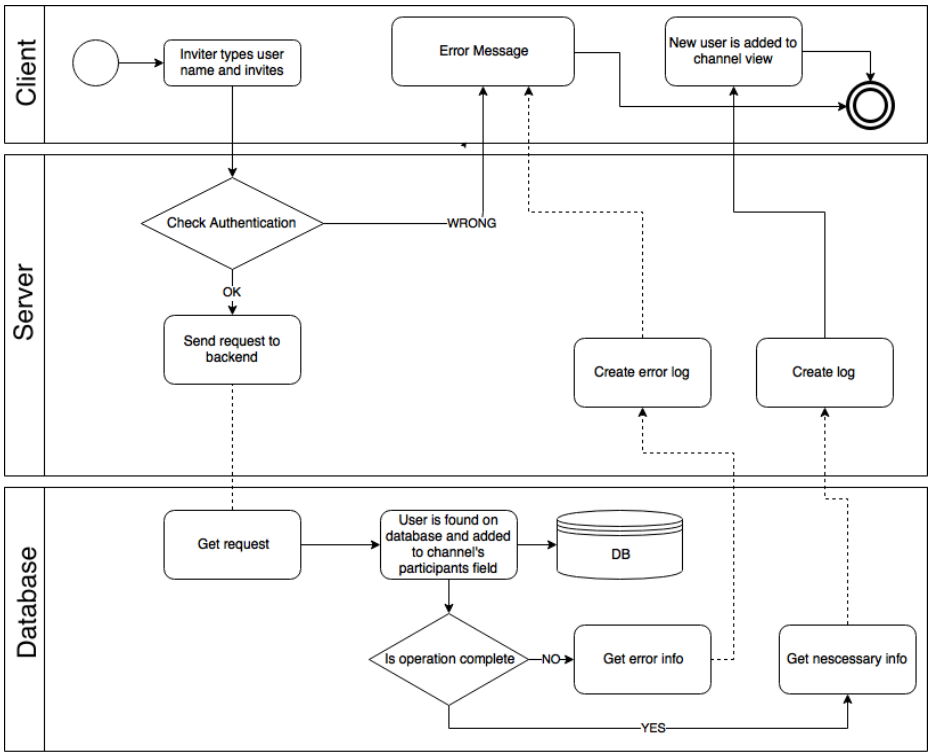


Figure 7 - Invite User

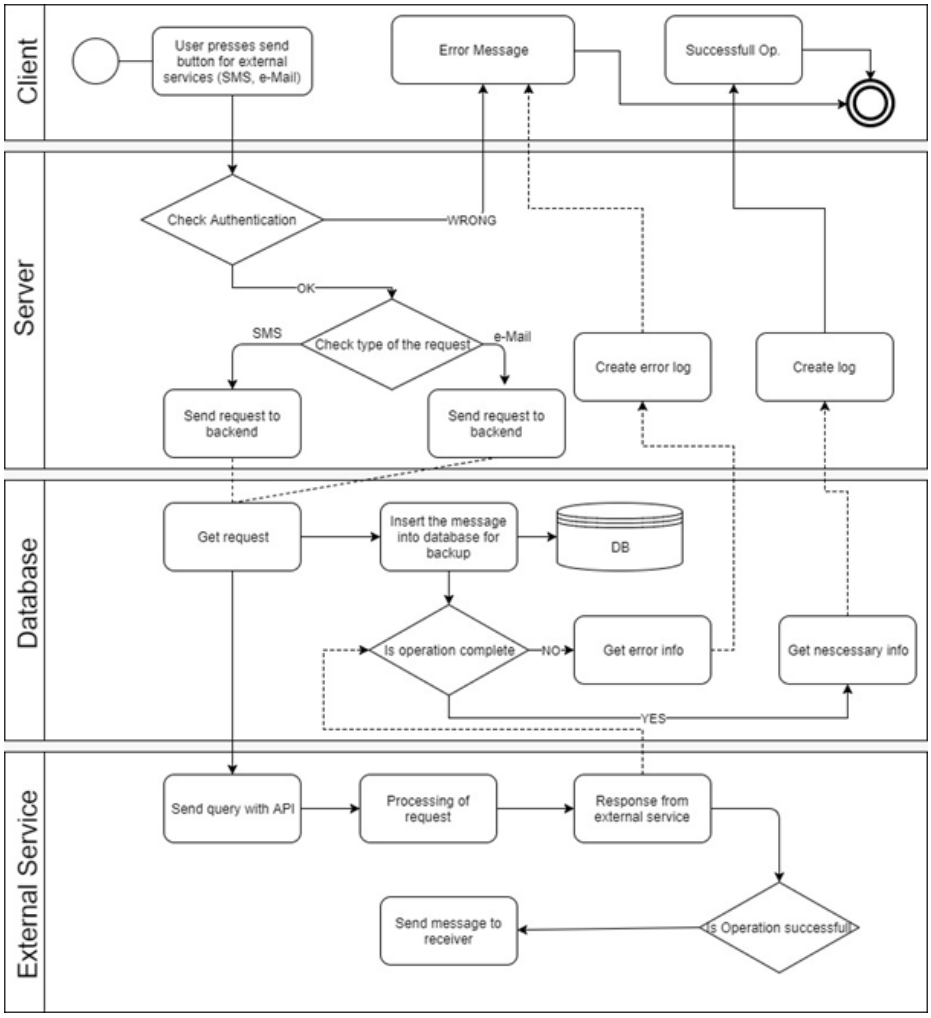


Figure 8 - Send Message via External Service

3.1.2. Class Diagram

Figure 9 depicts the general class diagram of the project built in consideration of object-oriented principles (cohesion, coupling, modular design). The diagram is constructed in the light of use cases and flow diagrams provided in early stages. Controllers, repositories, and domain classes are highlighted with different colors. Getters and setters are omitted to provide a clear overview.

In the development phase, we figured out that separating controllers with specific actions would provide a better workspace to avoid conflicts and improve cohesion. Furthermore, this approach would supply easier maintenance and development conditions in the future.

As the process of sending SMS and e-mail messages takes some time, and hence, blocks responsive user operations, an external message service is planned to be implemented with asynchronous methods. In this manner, SMS and e-mail messages will be queued and sent separately.

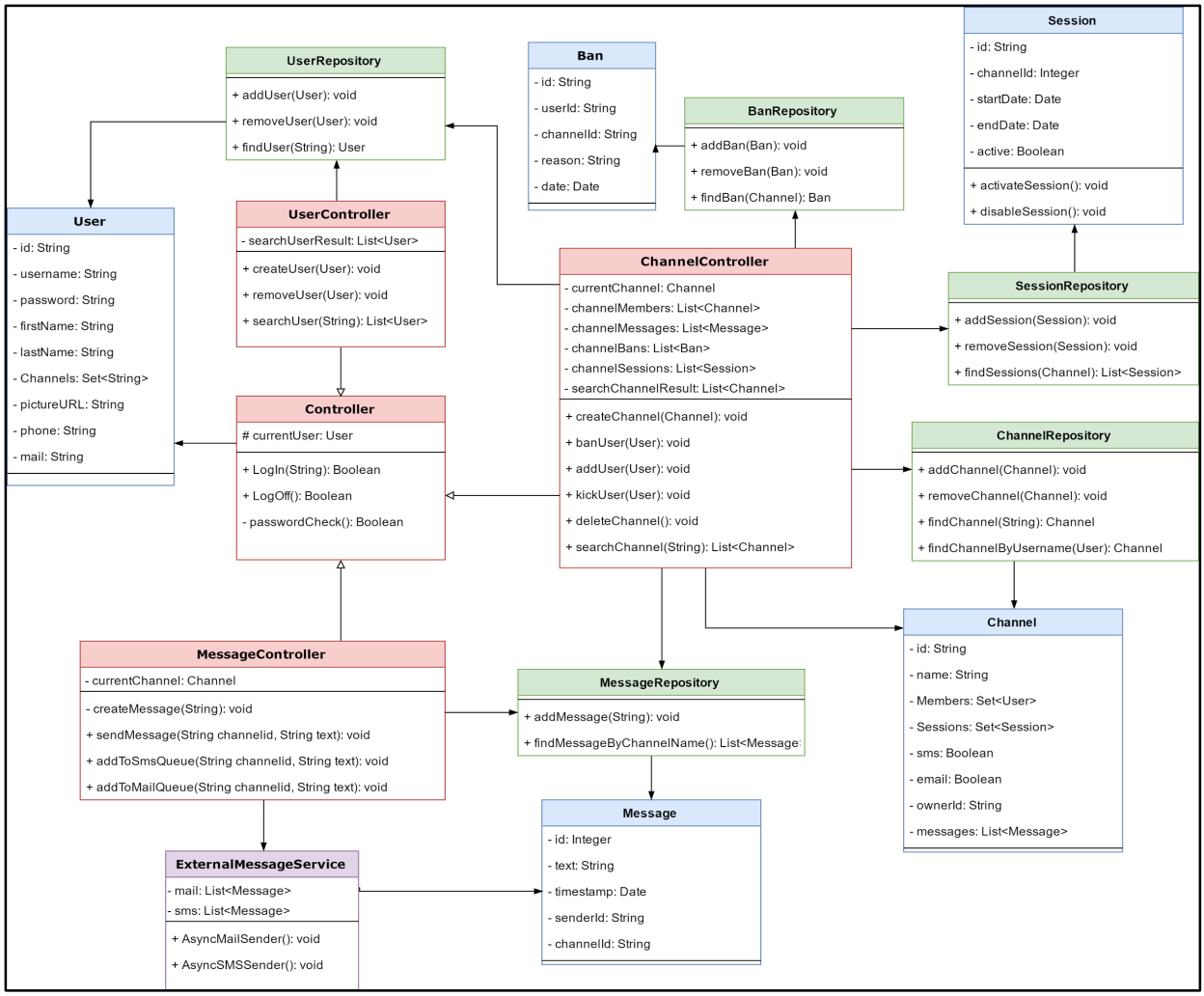


Figure 9 - Class Diagram of the system

More readable version of this class diagram can be reached via https://goo.gl/Za5dec.

3.1.3. UI Design Diagram

User Interface Diagrams are created for inspecting front end design issues in a broader perspective by determining necessary UI components, their members and relationships between components.

Project has two main user interfaces, landing page and main layout. The landing page lets users to login and register to the application.

The main layout is the basic UI of the project. It contains SideBarUI and NavBarUI. SideBar and Navbar UI provide navigation through the site and other functionalities like sign out. To be more precise, NavBarUI lets users to navigate their profile, delete their profile and sign out from the system. On the other hand, SideBarUI lets users to search for channels or users, provides routing to landing page, profile page, info page, schedule page, favorite messages page and create channel page. It also lists user's channels and joined channels.

Remaining UI's extends the main layout. UserProfileUI provides information about profile's owner, user's channels and schedule. Additionally, if the user owns the profile, UI lets the user to route CreateNewChannelUI. ScheduleUI shows user's schedule on a calendar with three different projection styles: monthly, weekly and daily. CreateNewChannelUI lets users to create new

obfuscated communication channels and select availability periods for these channels. FavMessagesUI lists user's favorite messages. ChannelUI lets users to send messages, add messages and channels to their favorites and remove them from their favorites. UI also provides additional different services according to different user roles, channel owner and channel member. MemberUI gives the opportunity to leave a channel to members and OwnerUI lets users to ban, unban, kick, add other users and delete the channel. SearchUI lists user's query results.

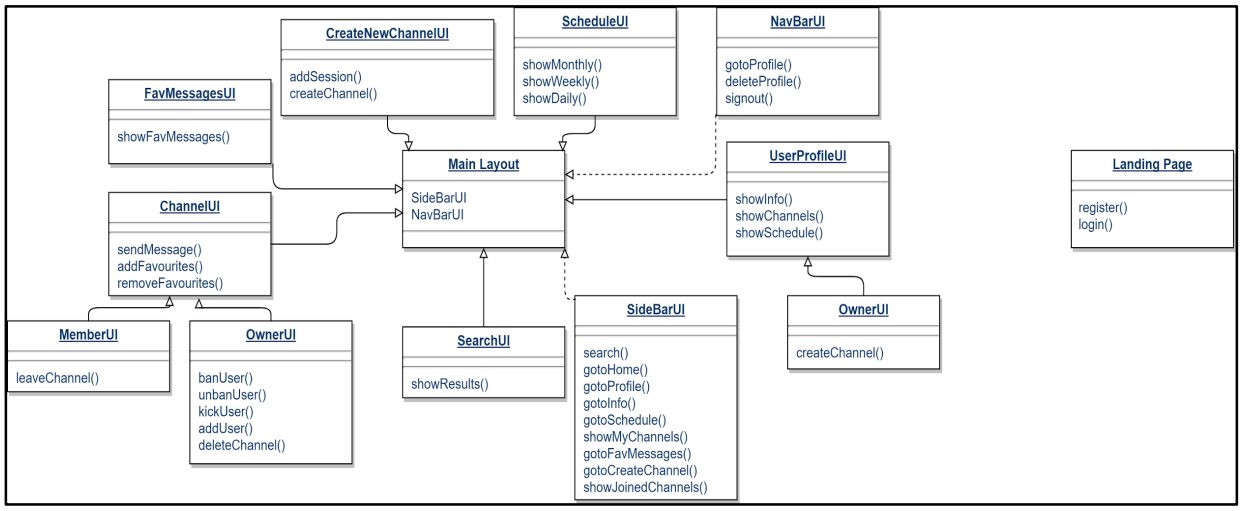


Figure 10 - UI Design Diagram of the system

3.1.4. Sequence Diagram

Figure 11 shows the typical sequence of events that occur while sending a message to a channel which may include SMS or mail options.

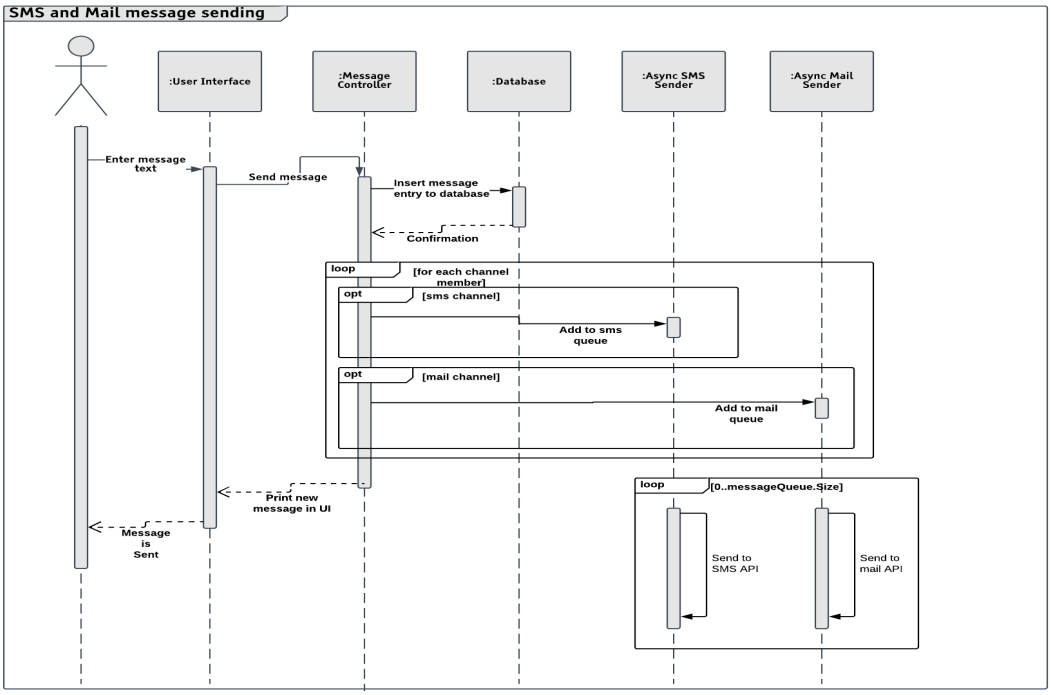


Figure 11: Sequence Diagram for sending messages

3.2. General Data Model

Figure 9 portrays the data model of the project as an E-R diagram. Although the application is developed using a non-relational database (MongoDB), relations between classes were designed as shown. As shown in Figure 9, there are four main data sources: User, Message, Channel and Session. A channel is associated with a session with “Create” relation, and “Hold” its messages. A channel is also associated with a User, the user’s rights to enter the channel, and the user’s favorite messages, channels.

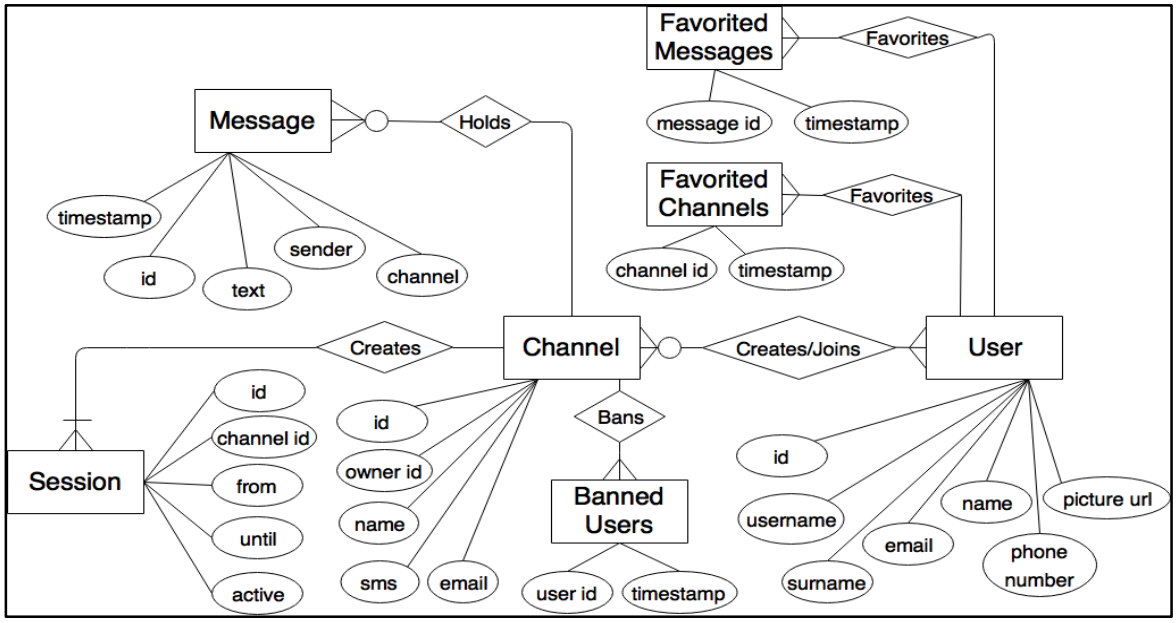


Figure 12 - General Data Model

4. Description of Implementation

In order to analyze the project, development environment, tools that are used and their dependencies should be closely examined. According to team’s common view, Java EE 8 was chosen as a platform [3]. Since time is limited due to strict course plan, the key criteria was to choose a robust platform that all team members could quickly develop and deliver the main functionalities of the application. Accordingly, Java EE was the perfect choice with respect to the team members’ prior experience. Besides that, platform’s high scalability and extensive support were the other key factors affecting our decision.

The chosen framework for development was Spring framework 1.5.1 with Boot and MVC options [4]. Java EE applications tend to contain excessive amounts of "plumbing" code which makes development, maintenance and the most importantly unit testing time consuming. Spring framework with boot and MVC provides dependency injection and inversion of control which procures developing loosely coupled application. By this, unit testing would also become easier.

As a built mechanism, Maven is chosen over Gradle [5]. Based on the fact that Gradle requires groovy, it would bring extra burden on team. On the other hand, Maven uses simple XML format and also running integration and unit tests are faster and easier.

Choosing among a variety of Java servers were challenging; however, Tomcat 8.5[6] delivers modularity and less resource use. Therefore, we chose Tomcat as our Java server.

Since Eclipse Oxygen is fully integrated with Spring and provides a large variety of tools, it is selected as the development environment [7].

Git-based source control tool GitHub was used for version control. In order to avoid conflicts, developers worked in parallel on different components.

As this project is a messaging application, the amount of data is expected to grow rapidly and become considerably big data. In order to store and manage this text data, MongoDB 3.4.10 was chosen [8].

The responsive and user-friendly design was one of the main requirements. In that manner, AdminLTE was included in the project [9]. This open-source front-end template application contains lots of useful components which support both web and mobile applications.

The ChannelX application is heavily dependent on SMS and email features. With the aim of providing secure and reliable service, dependable API's were used such as Twilio for SMS service and Google SMTP for email feature [10]. Twilio is a widely used API provider for exchanging text and pictures. Whilst Google SMTP is the most popular choice for mail routing [11].

5. Verification and Validation

5.1.Testing Strategies

As the integration testing strategy, Big Bang approach was chosen considering the fact that this type of testing requires very little planning and the project is a small-sized [12]. Due to the application's class model, all units must be present, because most of the modules are dependent on others to function properly, and hence, completing all the modules before testing was the rational choice.

Big Bang approach's major drawback is the difficulty of detecting point of origin of the error; however, as the architectural choice of the project is simple, we think it would be easy to detect errors. Consequently, by considering the team's limited time, Big bang approach was selected. White-Box and Black-Box approaches were also used as structural and functional testing.

In White-Box testing, code coverage and application logic was inspected. With Black-Box testing, modules' expected outputs for the correct inputs were tested. For testing the units properly, it is decided that these two approaches should be used together since each approach tests different aspects of the code.

5.2. Unit tests

So far, Unit tests in the project were completed. JUnit was used and high coverage rate was achieved. Reasons for low coverage on some files are API functions that could not be modified as well as some void methods. But these files have no functional problems. For Black-Box testing, Mockito was used and following coverage rates were received.

services	76,4 %	318	98	416
SessionHandler.java	0,0 %	0	98	98
AsyncMail.java	100,0 %	21	0	21
AsyncSms.java	100,0 %	22	0	22
BanServiceImpl.java	100,0 %	39	0	39
ChannelServiceImpl.java	100,0 %	33	0	33
FavChannelsServiceImpl.java	100,0 %	38	0	38
FavMessagesServiceImpl.java	100,0 %	38	0	38
LogServiceImpl.java	100,0 %	28	0	28
MessageServiceImpl.java	100,0 %	33	0	33
SessionServiceImpl.java	100,0 %	33	0	33
UserServiceImpl.java	100,0 %	33	0	33
domain	94,9 %	921	49	970
Message.java	92,9 %	224	17	241
User.java	90,8 %	139	14	153
Channel.java	92,9 %	157	12	169
Email.java	93,4 %	85	6	91
Ban.java	100,0 %	63	0	63
FavChannels.java	100,0 %	33	0	33
FavMessages.java	100,0 %	33	0	33
Log.java	100,0 %	40	0	40
Session.java	100,0 %	123	0	123
Sms.java	100,0 %	24	0	24

Figure 13 - Unit Tests Coverage rates

5.3. Integration tests

As of now, Integration tests are still being worked on for Controller classes. More comprehensive versions of the integration tests are being constructed to analyze data interaction that is happening between modules. These tests are expected to finish by the end of January 2018.

5.4. Acceptance test

Based on use cases, acceptance tests were generated. All tests are manually executed on the UI, and the actual outcomes are compared against the postconditions in use cases.

Test Case	**Related Use Case**
Ability to perform channel or user search	1
Ability to join channel	2
Ability to create channels	3.1
Ability to set desired time intervals to channels	3.2
Ability to send in-app message	4
Ability to favorite channel or messages	5
Ability to invite user to channel	6

Ability to kick user	7
Ability to ban user	8
Ability to delete channel	9
Ability to send message via external service (SMS or e-mail)	10

Table 4 - Acceptance Testing Cases and related Use Cases

5.5. Additional Tests

In addition to our existing tests, some other tests are planned for a complete testing process. Based on the requests of the project sponsor, more tests for checking the memory usage, launch time of the website, and other performance related issues could be applied. We will contact with the project sponsor to clarify these issues.

5.5.1. Security Tests

ChannelX is basically an obfuscated communication channel and security plays a crucial role. In order to analyze vulnerabilities of our application, we plan to utilize several tools and approaches such as Vega for SQL Injection, manual checking for Log Injection, and Spring for Authorization Testing. Performing each of these tests is one of the currently worked areas to protect the system against malicious attacks.

6. Evidence of Implementation

Since the beginning, the Hermes team regularly used GitHub online repository to store and keep track the development of the project. GitHub not only provides the team a shared workspace but also reports a concrete evidence of the development process in detail. All the related commit history of this project can be accessed via the following link:
https://github.com/hermesanonymousmessaging/hermes.

In addition to GitHub, four short featurettes were recorded. In these videos, new user registration, channel creation, invitation to a private channel, sending and receiving messages, banning a user, unbanning a user, kicking a user and finally deletion of a channel can be seen. These are not all but some of the features that are implemented. You can access these featurettes from the links below.

- ➔ https://youtu.be/xrwGU1dtCKg
- ➔ https://youtu.be/C85-ENVncac
- ➔ https://youtu.be/Mgm0GidsLqA
- ➔ https://youtu.be/4ofLr3EsMOE

7. Post-mortem Analysis of the Project

After the project has been delivered, we analyzed our team's productivity and efficiency in accordance to our plans, and reported below:

- **Planning of Project**

  Tasks, milestones, work packages, deliverables and all stages of the project were planned after the team was formed. Then the information needed to manage the resources for the project was estimated accordingly. Resources, timing, and distribution of tasks were also organized using that estimation.
- **Documentation of Project**

  All phases of the project were rigorously documented for informing all the project stakeholders. Planning and designing phases documented for completing the project successfully using that document for project evaluation. Project Proposal, requirements, project design, testing of the project were documented during project life-cycle. Documenting all phases increased the traceability of the project from requirements till testing phases. Detailed documentation in requirements and design phases also eased the implementation significantly.
- **Timeline of the Project**

  Projects tasks, milestones and time schedule of the tasks were planned and visualized using Gantt chart, and all the milestones were successfully delivered in predefined time.
- **Implementation of the Project**

  Implementation of the project started after the design part is completed. Firstly, user interface of the application is implemented; then database and the back-end components were added. Implementation is done with help of all members of the team. In the development process, team members also contributed to this process by working in jobs outside of their planned duties.

  Additional specifications of the project are not implemented because main properties of the project was just completed within the predefined time, and there was no extra time for developing additional specifications.
- **Testing of the Project**

  Testing part of the project started with unit testing done by all team members, using JUnit. All service classes are successfully covered and tested with unit tests. After that integration test was completed for database classes and controller classes of the project using Mockito Tool. Testing part is completed after white-box testing of the project. We were planning to cover security vulnerabilities and performance of the application before the project's final deliverable deadline, but these tests have been left to January 2018 to meet the deadline of the SCORE.

8. Outcomes and Future Development

As mentioned earlier in the report, project's success criterion is delivering a functional, reliable and documented web application for the customers which provides anonymous transient shared communication channels. Some difficulties arose while trying to fulfill this criterion. Time scarcity was the main problem because of the tight schedules of team members. Some technical but not functional downgrades happened due to lack of time. Implementation and testing phases could not be maintained in parallel due to the need of meeting the course's deadlines. Furthermore, some misunderstandings within the team happened which caused revisions during the implementation; this took extra time. Nevertheless, despite all these problems, key accomplishments were achieved. First of all, technology selections directly affected our achievement; ease of use of technologies and team's prior experience on these allowed very fast progress. The specifications within the project scope were accomplished and all deliverables mentioned in the project charter were delivered. Also, the scope was managed with success, project team adapted to the changes requested by the customer quickly while fulfilling the project constraints. As a result, the delivered project was considered as a success.

At the end of the project period, the team gained valuable experiences. Next time, in the light of these experiences, deadlines will be considered with great attention while specifying project scope. The design phase will be kept longer and the team will discuss it more. Coupling and cohesion aspects will be considered during parallel design and development using incremental approaches.

Although the project has been completed substantially, there are still some post-project tasks and future considerations which are listed below. Finally, after adding these functionalities, regression tests should be applied in addition to the remaining planned tests mentioned in the section 5.5.

- **Free-text search for messages:** An elasticsearch server may be connected to the application to allow this functionality.
- **Group/Direct chat option:** This option can be added as channel property. By allowing this option, channel members may observe all channel conversation (group chat) or, send and receive messages only with the channel owner (direct chat).
- **Channel invitations:** Channel owners may send invitations, that can be both rejected or accepted, to users for adding them to their channel.
- **Channel join requests and channel privacy:** According to a privacy property; the channel being hidden, private or public; channels may not show up in the search results. Additionally, users may apply to channel owners for joining to the channels.
- **In-app notifications:** A notification bar can be supplied for users to see if there are any unread messages in their channels.

References

[1] SCORE Project: ChannelX. Retrieved January 15, 2018, from http://score-contest.org/2018/projects/channelx.php

[2] T. Lethbridge and R. Laganiere, Object-Oriented Software Engineering: Practical Software Development using UML and Java, McGraw Hill, 2nd edition, 2005.

[3] Java(TM) EE 8 Specification APIs. (n.d.). Retrieved January 15, 2018, from https://javaee.github.io/javaee-spec/javadocs/

[4] Spring Boot Reference Guide. (n.d.). Retrieved January 15, 2018, from https://docs.spring.io/spring-boot/docs/1.5.1.RELEASE/reference/htmlsingle/

[5] The Apache Software Foundation. (n.d.). Getting Started with Maven. Retrieved January 15, 2018, from https://maven.apache.org/guides/index.html

[6] The Apache Software Foundation. (n.d.). Apache Tomcat 8. Retrieved January 15, 2018, from http://tomcat.apache.org/tomcat-8.5-doc/index.html

[7] Eclipse documentation. (n.d.). Retrieved January 15, 2018, from http://help.eclipse.org/oxygen/index.jsp

[8] Release Notes for MongoDB 3.4. (n.d.). Retrieved January 15, 2018, from https://docs.mongodb.com/manual/release-notes/3.4/

[9] AdminLTE Documentation. (n.d.). Retrieved January 15, 2018, from https://adminlte.io/themes/AdminLTE/documentation/index.html

[10] Twilio Messaging API. (n.d.). Retrieved January 15, 2018, from https://www.twilio.com/docs/api/messaging

[11] Send email from a printer, scanner, or app. (n.d.). Retrieved January 15, 2018, from https://support.google.com/a/answer/176600

[12] D. Galin, Software Quality Assurance: From Theory to Implementation, Pearson Addison Wesley, 1st edition, 2004.