Overview

With the development of technology, it is much more convenient for people to communicate, even they are far away from each other. Furthermore, as mobile phone becomes more and more popular, the demands of chatting applications increase dramatically. Thus, our team has set out to develop a product to realize communicating function: Sup? A desktop application that could send messages.

Our project allows users to realize 1-on-1 conversation via simple messages. Those conversations should be performed in a secure environment using proper protocol. Meanwhile, we would like to design native client GUI for our users and our development programming language is Java.

Our intent is to functionalize the plain text first, and then add more features, like security and chat history, to this application. We plan to use a client/server model to offer our application to a widely distributed audience, so that it could be utilized worldwide in more circumstances.

Team Roles and Responsibilities

In our first meeting we established roles for each team member. Our team will operate with a fairly flat structure, in the sense that we don't plan to recognize a lot of hierarchy and intend to share many responsibilities. Even though we plan to share, each role has particular focuses. Those roles and focused responsibilities are as follows:

- Steve Jarvis - Project Lead
    - Responsible for organizing and assigning tasks, managing the software repository and continuous integration.
- Ismail Zoubi - Backup Project Lead, Developer
    - Will fill in for Steve in the case where Steve is unable to serve as the lead. In normal day-to-day activity operates as a developer.
- Colin Horowitz - Software Architect
    - Designs and documents the overall structure of our software and the high level interactions between components.
- Mubing Liu, Ye Liu, Zhen Chui, Daniel Alvarez - Developers
    - The primary responsibility of developers is designing the actual implementation of software components, writing the source code, and creating unit tests to assure quality.

Communication

Our group also determined that the best form of communication for us, at least for the time being, is simply email. As components are determined and work begins smaller working groups can meet as needed. Scheduling conflicts will likely prevent the entire team from meeting on a regularly scheduled basis.

Project Details

1. Project Name: Sup?
    a. Name is tentative and subject to change
2. The goal of the project is to provide an online chat application. It should work on several major operating systems allowing users to effortlessly chat with each other.
3. Major functionalities and requirements
    a. 1-on-1 conversations via text through a variety of devices.
        i. Chatting the core functionality of the program, and will be the first priority.
        ii. Conversing with individuals means the program also has to find those individuals online, meaning a contacts list feature is essential.
        iii. Initially the devices will be limited to computers, likely Windows and Mac OSes, but enabling other operating systems and mobile devices to chat using this program is a long term goal (likely longer term than just a semester).
    b. Native client GUI
        i. Essential to the success of any program, and especially chat programs, a functional and intuitive GUI needs to be implemented.
        ii. The GUI should be usable by users of all ages and technical abilities.
        iii. Longer term, formatting of text as a user preference will be implemented. Initially the user will be limited to a default font.
    c. Distributed, anywhere in the world
        i. This folds back on point a, where the software should function on a variety of devices. It also means the program has to be able to communicate across the internet, and not just on LAN or other local networks.
    d. Secure conversations
        i. Security is a major goal of any communication software, and after the basic communication framework is implemented, encryption will be worked on in ensure private conversations.

4. Minor functionalities and requirements - The “if time permits” features
    a. Chat History

        i. Saved either locally by the client or on a central server, the program should save chat histories so users can access old chat logs. If this is done centrally, database security becomes an important program feature as well.
    b. Chat Rooms
        i. Users sometimes prefer communicating with a group of friends at the same time, rather than in several 1 on 1 conversations.
    c. Status Settings
        i. Instead of simply “Online” or “Offline,” more variety in statuses allows users to know when to expect a response, or simply leave a message
    d. Anti-Spam features
        i. Namely block lists. It is important to be able to block messages from certain users. These messages usually consist of spam, but can also include hateful speech or unwanted conversation.
        ii. If a user is sending out messages too quickly to be human, that user can be timed out or even banned.
5. Hosted on GitHub and written in Java

Major Project Risks

1. We don’t all know the same language.
    a. We decided on Java, because it at least uses a general structure that’s familiar to all members of our team. Because of the familiar semantics we think the ramp up time can be minimized.
2. The tools are new to us.
    a. The ideas of version control, continuous integration, and other important components of Software Engineering will be new to us. We have some familiarity, and think these complications can be mitigated by dealing with them from the beginning.
3. No real experience making a graphical UI.
    a. Most of our programming experience has been back-end development.
4. Protocols in distributed applications can grow complex.
    a. The communications and design of components will be a crucial step to avoid major complexities and road blocks later on.

High Level Schedule

	October	November	December
Week 1	Planning Phase		

Week 2	Major functionalities a		
Week 3	Major functionalities a+b		
Week 4	Major functionalities a+b+c		
Week 5	Initial Version		
Week 6		Major functionalities b+c	
Week 7		Major functionalities c	
Week 8		Intermediate Version	
Week 9			Major functionalities a+b+c
Week 10			Final Version

Quality Assurance Strategy

1. Test Design
    a. Tests will be designed as the code is being designed in order to ensure the software meets the specifications
    b. Tests will cover 3 levels in a pseudo-V model.
        i. Unit testing for individual pieces of the code, as each piece is being designed
        ii. Component testing for adding components to the software, as each component is being added to the software
        iii. Acceptance test to ensure entire program will meet the customer's demands for each version
2. Other Metrics
    a. Code Review
        i. On a weekly basis at least 1 other person of the team will look at the code developed to make sure it is both doing what it is supposed to, and easy to understand for facilitated maintenance.
3. How to measure those metrics
    a. Continuous integration
        i. The testing will be applied as the programmers add code to the GitHub repository
4. Code standards

a. Consistent variable usage and method calls
    i. Descriptive variables used for maintainability, and method calls kept reliable to easily use them without always referring to documentation
b. Documentation and Comment Prioritization
    i. Proper documentation created before coding begins, but will be edited immediately if code has to be changed for some reason
    ii. Effective use of comments to read and maintain the code
c. No formatting standards
    i. Code can otherwise be written as most comfortable to the programmer. Spacing and other formatting standards are nice to have, but can be burdensome to non-professional coders.

Related Work

1. **AIM**

   When it first released in 1997, AOL Instant Messenger(AIM) was the first generation of real time communication services. By downloading a small program on computer, one could send short text to more than one people at the same time. Furthermore, it enables users to send voice message even before Skype, and this feature has now been embedded in almost every messenger application.

   AIM rises and falls quickly. It attracts more than 53 million users by 2005 and the figure dropped to 4 million in 2012. Still It perceived as one of the pavement stone for future IM applications. For our project we would like to build the fundamental program first then provide additional features accordingly. The experience of AIM could serve as an excellent textbook for our early development.

2. **MSN**

   MSN was first launched by Microsoft in 1999. Due to its strong industrial influence, it attracted 330 million users by 2009 and its name was known to every avid IM user. In addition to its fundamental messaging functionality it also offered customers a way to “exchange online messages and email with the more than 40 million users of the MSN Hotmail”. It was renamed Window Live Messenger and start to falling by the wayside as other IM services rise such as GTalk, Yahoo IM, Digsby, Trillian, etc.

   One outstanding feature of MSN comparing to some old original IM apps is that it integrated multiple IM accounts into one client instead of making users feel that they are isolated by their friends from other social networks. Therefore, we find it’s quite necessary to take a deep look into MSN, since this compatible problem happens to almost every IM user, and integrating multiple IM accounts into one is a good solution to such problem.

3. **Whatsapp**

Founded in 2009 by some ex- Yahoo employees, Whatsapp really has taken off in recent years as a cross platform mobile app. Despite being less feature-rich than its main competitors such as Skype and MSN, Whatsapp acts as a substitute of traditional messaging service that is able to receive messages in the background.

One thing about Whatsapp we are interested in is that it allows users to add friends via scanning contact information while other IM app requires much more effort. By 2015, Whatsapp has acquired 500 million users which in turn makes it a widely used app for mobile device.

4. Wechat

Launched just 4 years ago by Chinese investment holding company Tencent, Wechat is the most widely used messaging app in China. Even though its popularity is yet class-leading in the global market, Wechat is no doubt a pioneer as a combination of platform and mobile portal. As Facebook and Whatsapp are linked with the growth of web and mobile messaging, and focusing on build the largest social network, Wechat on the other hand, is focusing on display every aspect in user's life.

Wechat, like Whatsapp, is not first born as a website then adapted to mobile phone, it was born of it. Similar to Whatsapp Wechat was build sorely as a mobile application instead of been transformed from a web-end tool. But compared to Whatsapp, Wechat provides more features including a new model "app within app". Millions of lightweight apps live inside Wechat, in which similar to the way webpage lives on the internet, which makes it more like a browser of mobile website. So if we finally want our project to be multi-used and commercial, WeChat model is a good point to start with.