Chase Brown

Software Design Specification

Server Client Personal Calendar

Revision 1.0

Table of Contents

Revision History

Version	Name	Reason For Changes	Date
1.0	*Chase Brown*	*Initial Revision*	*03/20/2018*

1. Introduction

1.1 Purpose

Design an on-line personal calendar using TCP sockets and developed protocol.

1.2 System Overview

This project is intended to create an on-line personal calendar using TCP sockets and a custom protocol. The calendar protocol shall be able to add a user, delete a user, modify user information, user authentications, add a new appointment, remove an appointment, update an existing appointment, display a user's appointments for a specific time.

2. Design Considerations

All design considerations were handled in phase 1.

2.1 Assumptions

Use of all C++11 functionality

2.2 Constraints

All programs shall be written in C or C++ and run on UNIX like platform.

2.3 System Environment

Required to run on CSEgrid machines

2.4 Risks and Volatile Areas

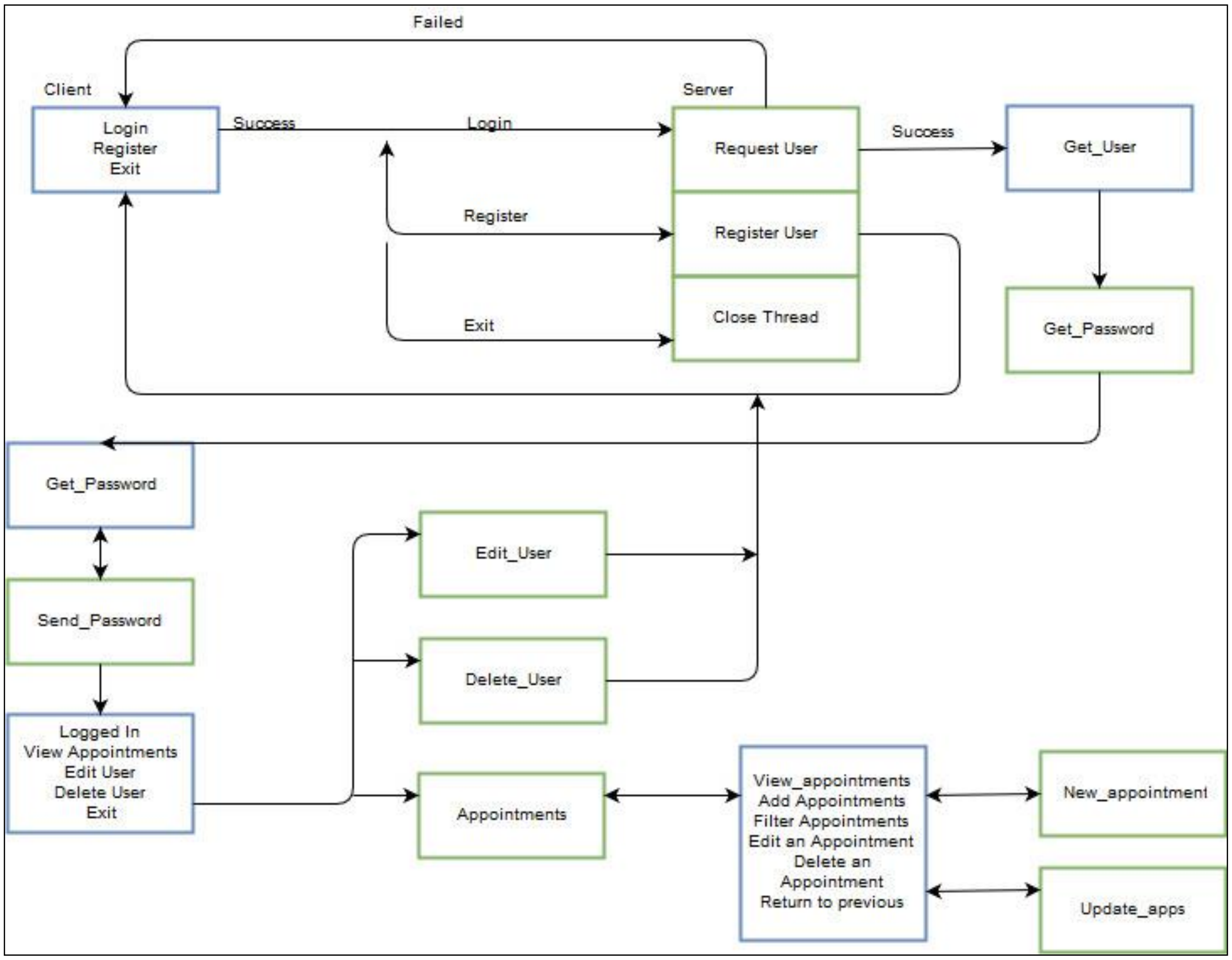
- Check a time conflict(duplicated appointments)
    - Appointments can be made at the same time and date as long as they do not have the same name. Modern calendar systems allow users to create events at the same time or overlapping.
- Check duplicated users
    - During registration of a user if the user exists in the database an error is returned, and client is sent back to login/registration.
- Check incomplete schedule
    - Schedule is forced to have all areas filled before it is sent to the server for storage.
- Check invalid time and character.

- ~~Inputs are checked via regex strings and not passed until valid.~~ Regex is not supported by the CSEgrid GCC(4.8.2). Regex strings require 4.9+ GCC. Commented out.
- Inputs are checked via functions and not passed until valid.

3. Architecture

The architecture provides the top-level design view of a system and provides a basis for more detailed design work

Provide or reference a detailed description and diagrams of the architecture.



3.1 System Architecture

Server: TCP socket server to be ran on a node awaiting connections.

Client: TCP socket client, able to run on multiple nodes connecting to a single server.

3.2 Sever Components

TCP Socket multi-threaded server written with C++
Ability to listen for new connections and create threads to handle such connections.

- Main.cpp
    - Main()
        - Server implementation/start
        - Connection threading
    - *connection_handler()
        - Handles all operations for thread created in main.
- ClientSocket.h/cpp
    - ClientSocket class definition
    - Message struct definition
    - Enum message_type
    - Enum header_type
- Fileops.h/cpp
    - Save_new_user()
        - Save
    - Look_up_user()
        - Check if user exists.
    - Get_user_appointments()
        - Retrieve the current user appointments
    - Save_appointments()
        - Save user appointments
    - Load_appointments()
        - Load user appointments
    - Save_updated_appointments()
        - Save user updated appointments
- Passwordfunctions.h/cpp
    - Random_string()
        - Generates a random string with length of user password
    - Decrypt()
        - Decrypt a string using XOR
    - Encrypt()
        - Encrypt a string using XOR
- Users.h/cpp
    - Appointment class definition
    - User class definition
    - string_is_valid()
        - Validates string is alphanumeric
    - password_is_valid()
        - Validates password is alphanumeric and of length 8-16
    - phone_id_valid()
        - Validates phone is numeric and of length 10
    - check_valid_time()
        - Validate time format
    - check_valid_data()
        - Validate date format

3.3 Client Components

TCP Client written with C++
Ability to connect to a server by a given IP.

- Main.cpp
  - Accepts client IP argument
  - Connects to Server
- ClientSocket.h/cpp
  - ClientSocket class definition
  - Message struct definition
  - Enum message_type
  - Enum header_type
- Passwordfunctions.h/cpp
  - Random_string()
    - Generates a random string with length of user password
  - Decrypt()
    - Decrypt a string using XOR
  - Encrypt()
    - Encrypt a string using XOR
- Users.h/cpp
  - Appointment class definition
  - User class definition
  - Create_user()
    - Creates new user, Name, Password, Phone, Email.
  - Edit_user()
    - Edit User info, Name, Password, Phone, Email.
  - New_appointment()
    - Creates appointment, start time, end time, start date, end date, place, details.
  - Ed_apps()
    - Edit current appointment
  - Del_apps()
    - Delete current appointment
  - Filter_apps()
    - Filter appointments for view
  - isValidDate()
    - Validates date
  - isValidTime()
    - Validates time
  - isValidTimeRange()
    - Validates time range.
  - string_is_valid()
    - Validates string is alphanumeric
  - password_is_valid()
    - Validates password is alphanumeric and of length 8-16
  - phone_id_valid()
    - Validates phone is numeric and of length 10
  - check_valid_time()
    - Validate time format
  - check_valid_data()
    - Validate date format

3.4 User Database

Simple directory ‘Users’ to hold .csv files for users and their appointments.

- User Files:
  - One file per user with name format: ‘username’.csv
- User Appointments Files:
  - One file per user with naming format: ‘username’appoint.csv

3.5 Protocol Design

Packets will be sent as a ‘message’
Allows Client and Server to communicate as expected.

- Message : Struct
  - Type: enumerated value
    - CONNECT_OK
    - TRACK
    - SUCCESS
    - FAILER
    - QUIT
  - Header: enumerated value
    - GET_USER
    - SEND_USER
    - GET_PASSWORD
    - SEND_PASSWORD
    - LOGIN
    - REGISTER
    - MENU
    - APPOINTMENTS
    - EDIT_USER
    - DELETE_USER
    - NEW_APPOINTMENT
    - VIEW_APPOINTMENT
    - MENU2
    - UPDATE_APPS
    - SAVE_UPDATE_APPS
    - GOOD
  - Payload: char[255]