TeamEditor

Contributors:

Pratith Kanagaraj

pxk5958@rit.edu

Yogeesh Seralathan

ys4815@rit.edu

1.

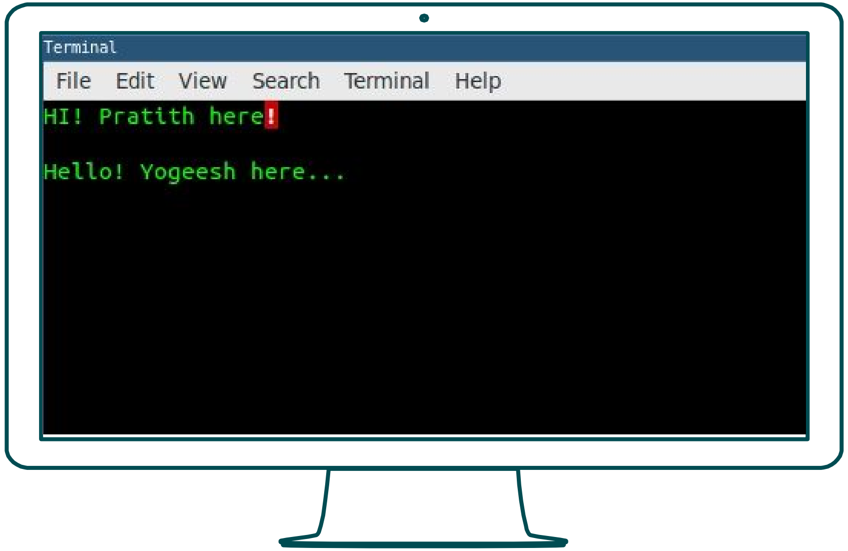
What is TeamEditor?

A real-time collaborative text editor!

Purpose

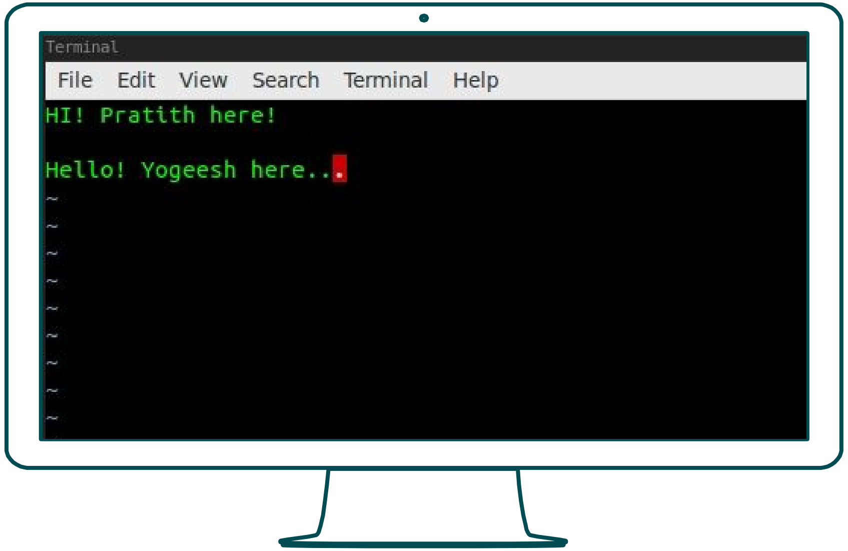
- Create documents simultaneously with your friends/colleagues (like this presentation)
- Edit/review code collaboratively in real-time

Result: Save time and effort



```
HI! Pratith here!

Hello! Yogeesh here...
```



```
HI! Pratith here!

Hello! Yogeesh here...
~
~
~
~
~
~
~
~
~
```

Demo time!

2.

Just a text editor?

Larger Systems

- Real-time chatting
- Collaborative Image editing
- Collaborative IDE
- Collaborative Circuit building
- etc.

Open Ended!

Any collaborative editing application you can think of

3.

Design

Design

- Model-View-Controller
- Client-Server
- JSON data
- Differential Synchronization

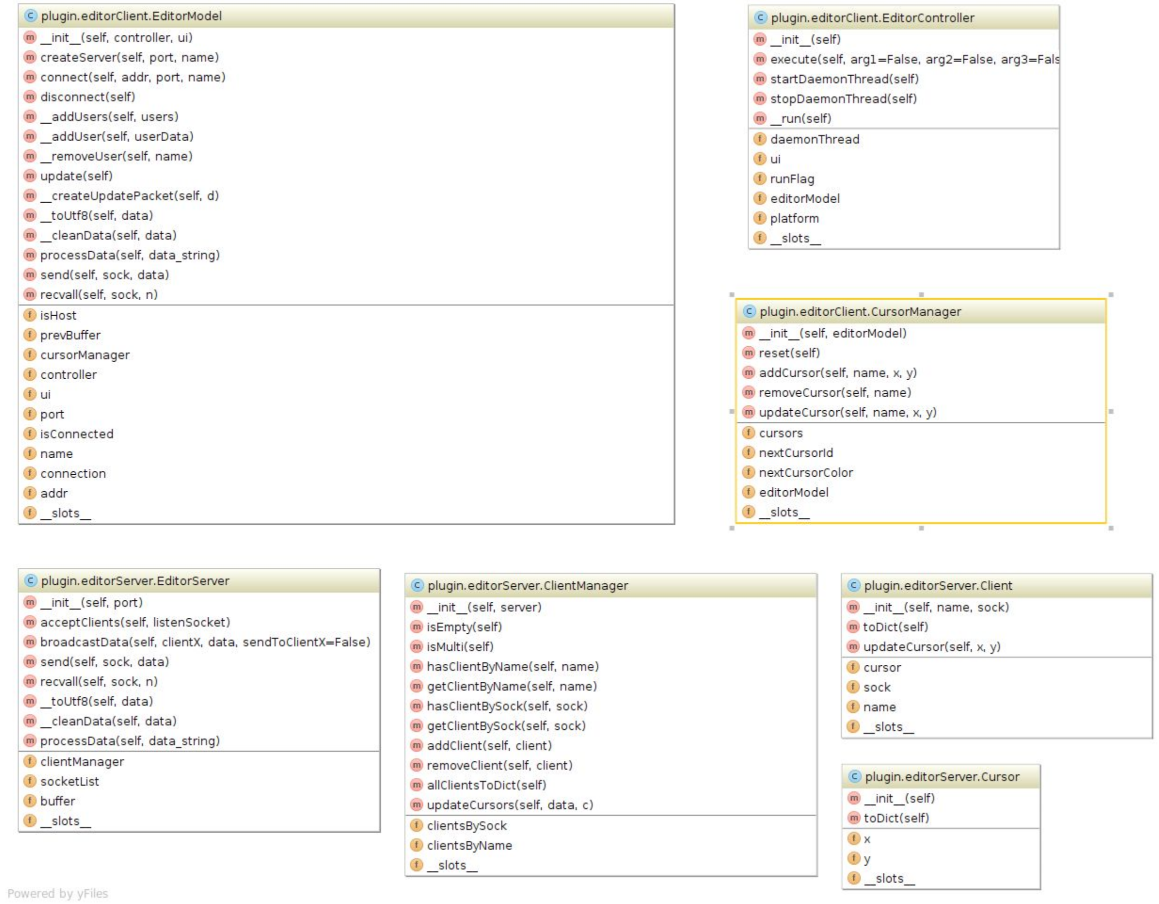
4.

Implementation

Implementation

- Vim as User Interface (the View)
- Python, VimL as programming language
- 11 colored cursors representing 11 users (more can join with repeated colors)
- Updates are polling-based

Client-Server UML



```
plugin.editorClient.EditorModel
  m __init__(self, controller, ui)
  m createServer(self, port, name)
  m connect(self, addr, port, name)
  m disconnect(self)
  m __addUsers(self, users)
  m __addUser(self, userData)
  m __removeUser(self, name)
  m update(self)
  m __createUpdatePacket(self, d)
  m __toUtf8(self, data)
  m __cleanData(self, data)
  m processData(self, data_string)
  m send(self, sock, data)
  m recvall(self, sock, n)
  f isHost
  f prevBuffer
  f cursorManager
  f controller
  f ui
  f port
  f isConnected
  f name
  f connection
  f addr
  f __slots__
```

```
plugin.editorClient.EditorController
  m __init__(self)
  m execute(self, arg1=False, arg2=False, arg3=Fals
  m startDaemonThread(self)
  m stopDaemonThread(self)
  m __run(self)
  f daemonThread
  f ui
  f runFlag
  f editorModel
  f platform
  f __slots__
```

```
plugin.editorClient.CursorManager
  m __init__(self, editorModel)
  m reset(self)
  m addCursor(self, name, x, y)
  m removeCursor(self, name)
  m updateCursor(self, name, x, y)
  f cursors
  f nextCursorId
  f nextCursorColor
  f editorModel
  f __slots__
```

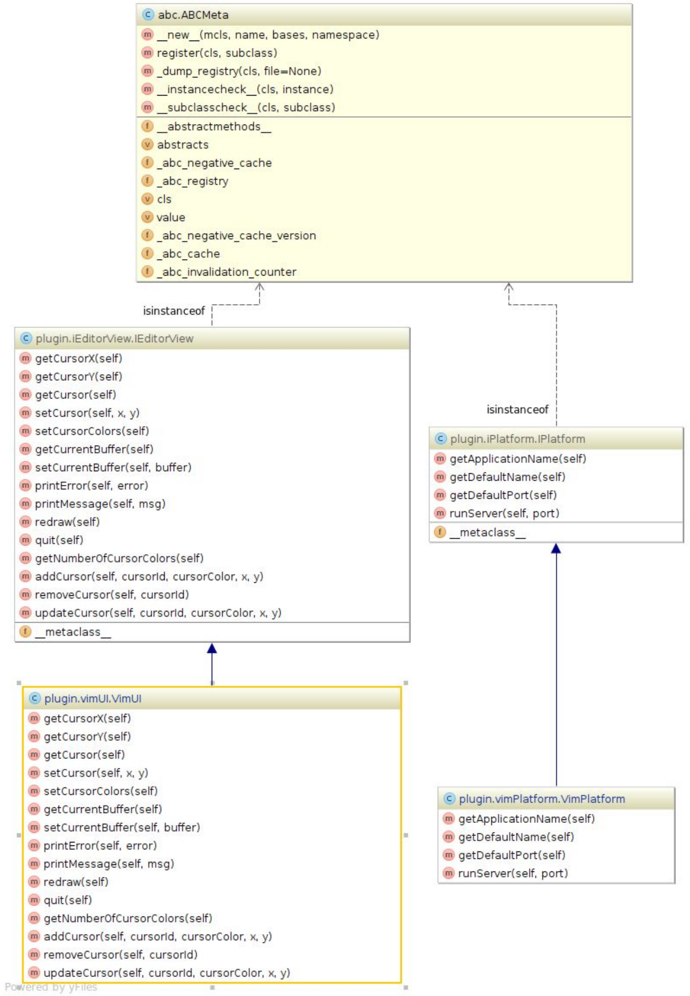
```
plugin.editorServer.EditorServer
  m __init__(self, port)
  m acceptClients(self, listenSocket)
  m broadcastData(self, clientX, data, sendToClientX=False)
  m send(self, sock, data)
  m recvall(self, sock, n)
  m __toUtf8(self, data)
  m __cleanData(self, data)
  m processData(self, data_string)
  f clientManager
  f socketList
  f buffer
  f __slots__
```

```
plugin.editorServer.ClientManager
  m __init__(self, server)
  m isEmpty(self)
  m isMulti(self)
  m hasClientByName(self, name)
  m getClientByName(self, name)
  m hasClientBySock(self, sock)
  m getClientBySock(self, sock)
  m addClient(self, client)
  m removeClient(self, client)
  m allClientsToDict(self)
  m updateCursors(self, data, c)
  f clientsBySock
  f clientsByName
  f __slots__
```

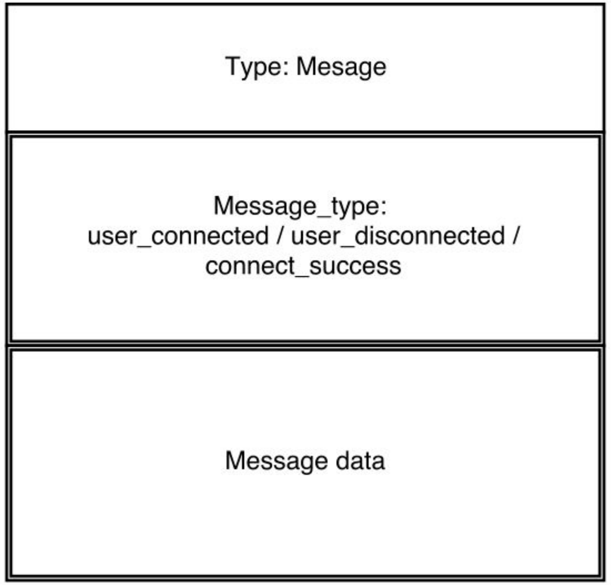
```
plugin.editorServer.Client
  m __init__(self, name, sock)
  m toDict(self)
  m updateCursor(self, x, y)
  f cursor
  f sock
  f name
  f __slots__
```

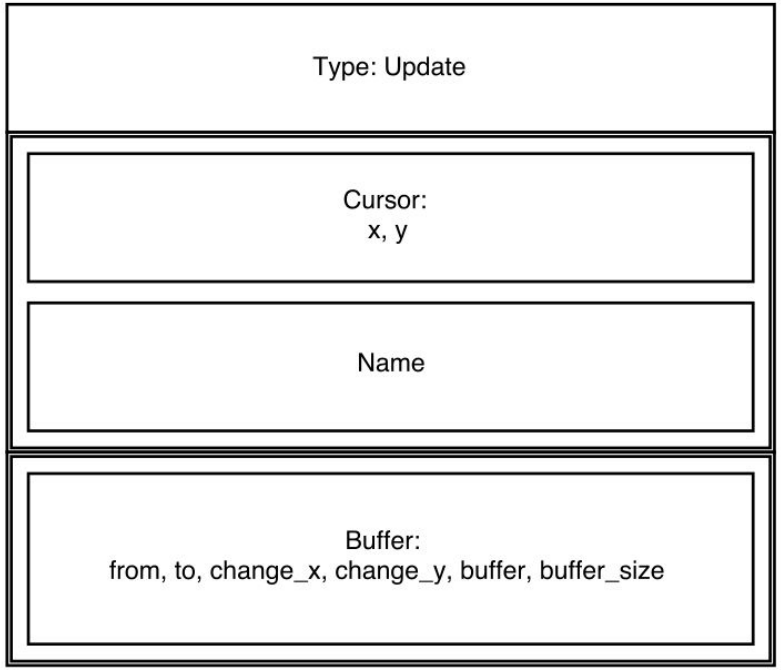
```
plugin.editorServer.Cursor
  m __init__(self)
  m toDict(self)
  f x
  f y
  f __slots__
```

Vim UI,Platform UML



Protocol


Type: Mesage
Message_type:
user_connected / user_disconnected / connect_success
Message data


Type: Update
Cursor:
x, y
Name
Buffer:
from, to, change_x, change_y, buffer, buffer_size

5.

Improvements

Improvements / Extensions

- Username validation
- Event-based updates
- Operational Transformations
- XML based approach for extensibility
- Encryption
- Authentication
- Unique Document ID
- Web interface

Roles

Pratith

Client implementation

Yogeesh

Server implementation

Thanks!

Any questions?

You can find us on twitter at @yogeesh93 and @pratith