Communication Tool Iteration #3

Group 1, Team 2:
Oscar Ingham, William Wilson, Shishuang Shu, Yuyan Zhang, Xinzhu Xu

Roles

Bill: Team Leader

Oscar: Implementation Leader

Jenna and Sherry: Front end & Functional Tests

Bill and Jewel:Backend Team

Presentation

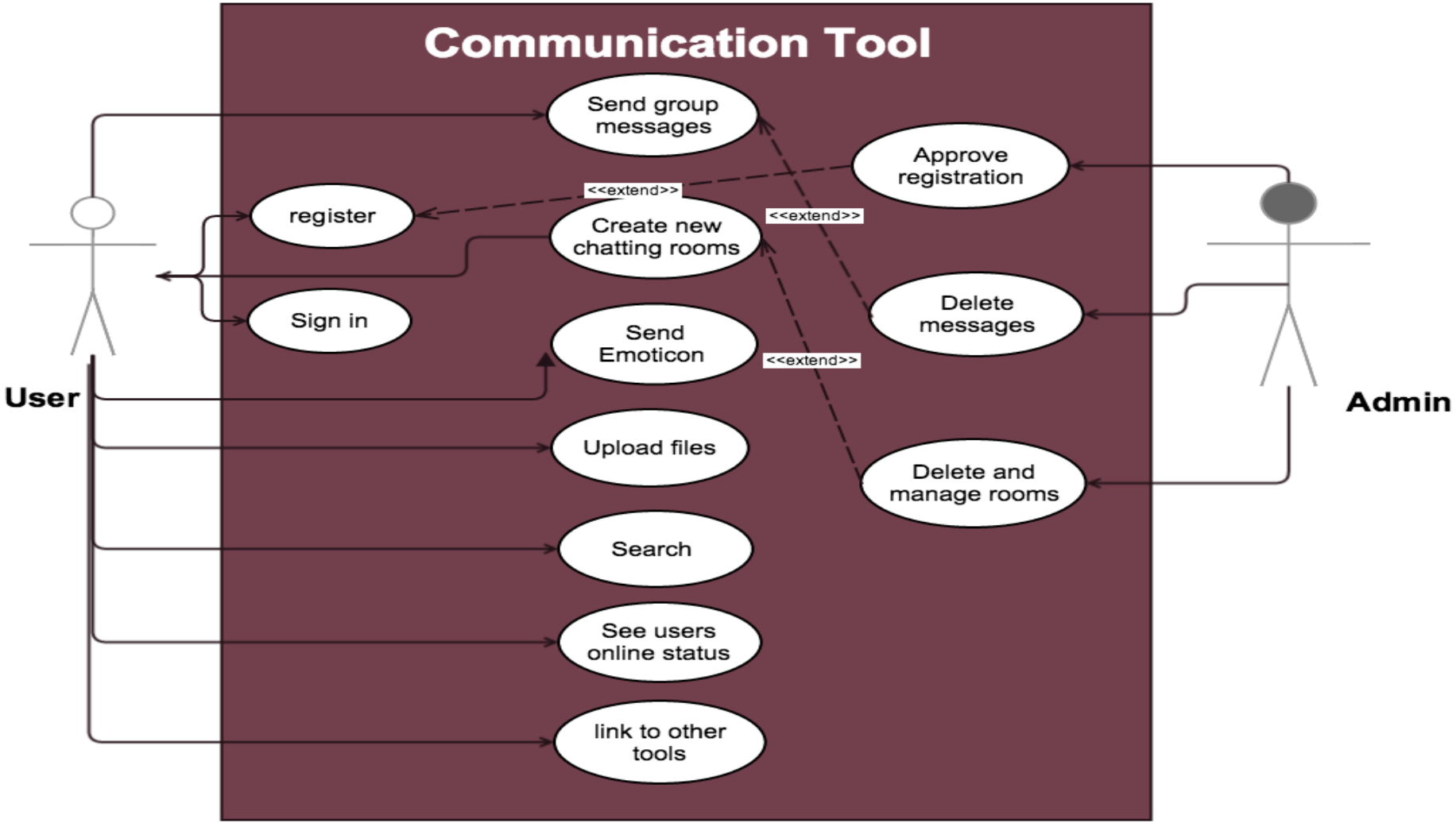
1. Requirements Analysis
2. Design/Architecture/Implementation
3. Testing
4. Risk Analysis
5. Quality Management

Requirement Analysis

Functional Requirements

- Use case diagram
- User story
- How to track requirement and handle change

Non-Functional Requirements


Communication Tool
Send group messages
Approve registration
register
<<extend>>
Create new chatting rooms
<<extend>>
Sign in
Delete messages
Send Emoticon
<<extend>>
User
Admin
Upload files
Delete and manage rooms
Search
See users online status
link to other tools

Requirement analysis-- User story

- Total number of user story is 19.
- With 13 completed and 6 uncompleted.
- Total point of user story is 86.
- With 72 points completed and 14 points uncompleted.

Requirement analysis-- User story

Eg:

Send Group Messages

As a user in a group, I can send text messages to the whole group publicly so that all the member in this group can view the conversation.

Requirement analysis-- User story

Eg:

Record Searching

As a user, I can search for the certain text in the conversation record as well as the sharing files.

How to track the requirement?

- Writing user stories in pivotal tracker.
- According to the priority,estimate scores for each user stories.
- Start to achieve the function with higher scores first.
- Keep updating the pivotal tracker about whether the user stories has been achieved or not.

Non-functional requirements

- User-friendly/Performance

Simple: easy and simple to use.

Responsive: The User Interface should be responsive according to the size of device.

Speed: low latency response to button click or message sending.

Project Requirements for Iteration #3

- User model Integration
- File Upload
- Record Searching
- Emoticons
- Unit tests

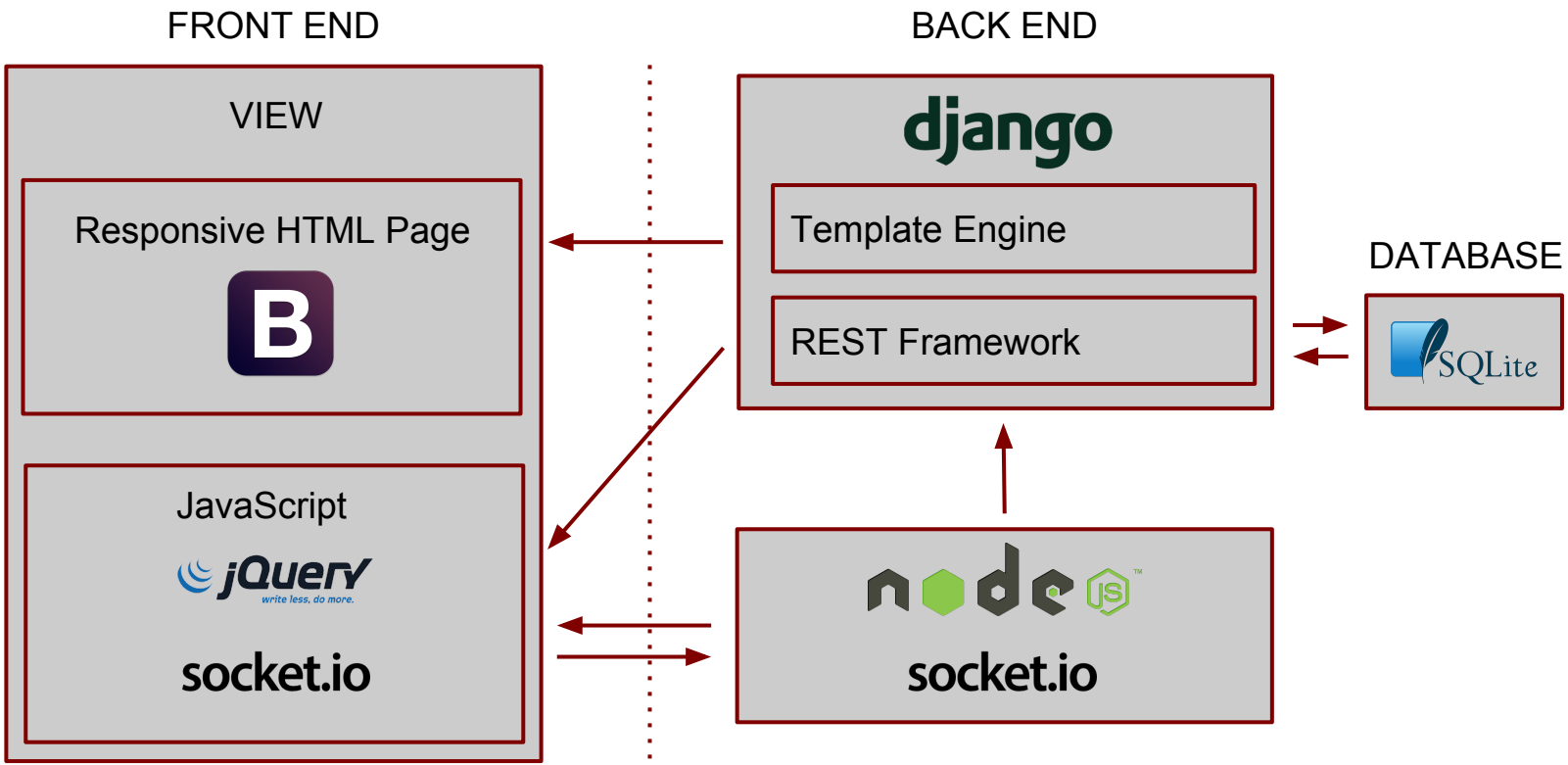
Design

- Real-time chat is an event-driven application
  - Django is built on a request-response cycle
    - Periodic polling would likely result in poor user experience
  - Alternatively, web-sockets allow persistent bi-directional communication between client and server.
- Design patterns
  - Model-View-Template (MVC-like)
  - Publish-Subscribe model (Observer)

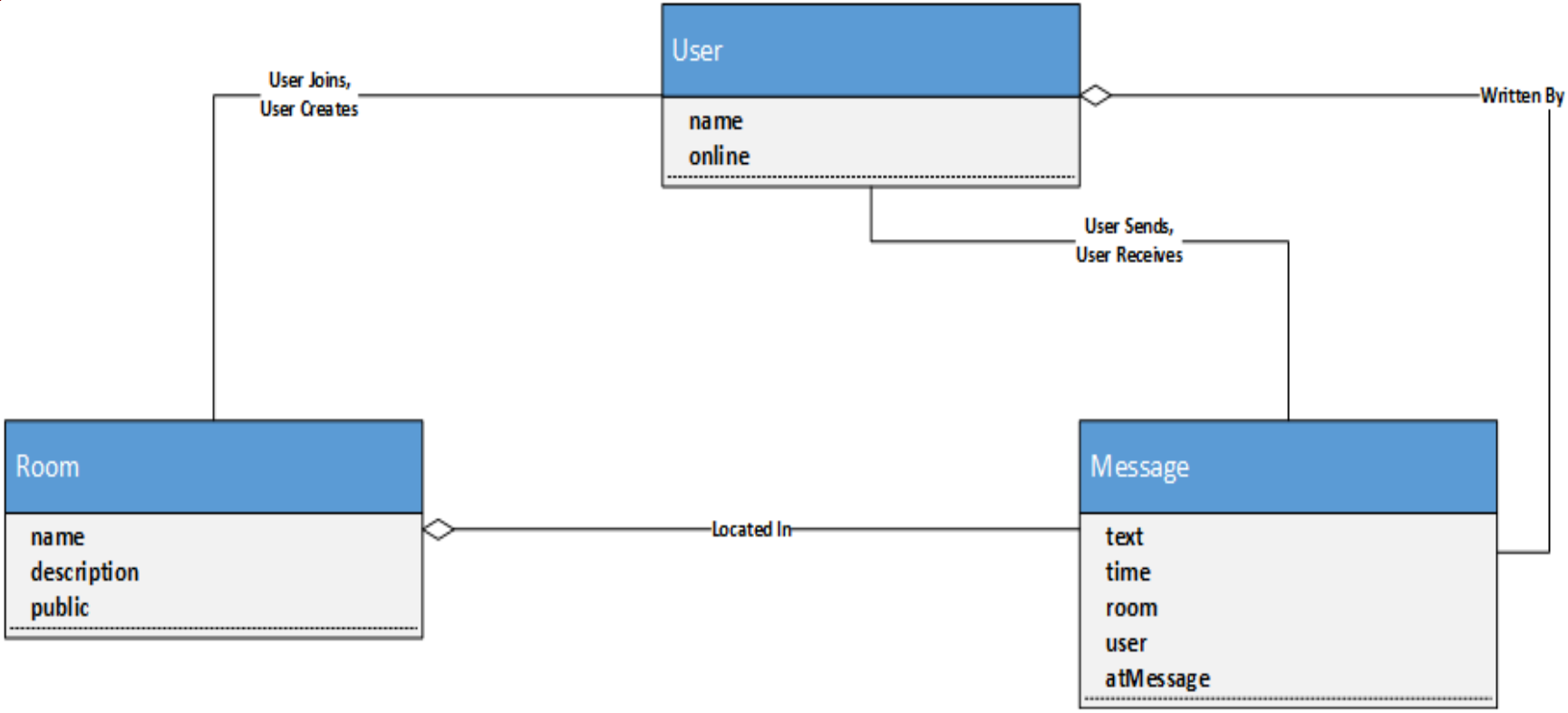
Architecture

- Single HTML template served through Django
  - Integrates Django user authentication mechanism
  - Responsive CSS templates with Bootstrap
- Data managed via AJAX and Django Rest Framework
  - JQuery mostly used on the front-end
- SocketIO for real-time messages
  - Separate NodeJS service (port 3000)
  - Manage collections of websockets by namespace
  - Dynamically create new sockets based on user input

Architecture



Class Diagram



Implementation

- IDE/Editor: Sublime Text / gEdit
- Node/Browser JS Interpreter & iPython
- Development within a VirtualBox VM

- Git for Version Control
  - One repo hosted on GitHub
  - Feature-branch workflow:
    - Created branches for each feature we were working on.
    - Once complete, user filed a pull request. Team reviewed the code, and merged into our master branch once accepted.

Environment

- Moved from Ubuntu to CentOS and eventually back to Ubuntu…
  - An attempt to keep our environment in development as close as possible to production.
  - An interesting learning exercise anyway.

- Gunicorn - WSGI server
- NGINX - Web Server, proxies requests to Gunicorn and serves static files
- Supervisor - Keeps Gunicorn and Node alive
  - Replaced with Upstart in production

Implementation - Back End

- Python
  - REST API for managing users, rooms, messages
  - REST endpoint for finding messages in the database
  - One Django view

- NodeJS
  - Manage collection of sockets via namespaces (SocketIO server)
  - Global namespace to handle general event announcements (i.e. user connected, new room created)
  - File upload
  - Show currently connected users

Implementation - Front End

- CSS:
  - Responsive layouts for different screens and devices
- HTML:
  - DOM elements (i.e. inputs, pop-up modals, messages and images)
- JavaScript:
  - SocketIO client to manage WebSocket connections
  - All AJAX via REST framework (jQuery)
  - DOM manipulation (i.e. changing header bar, adding new room) (jQuery)

Testing

- Unit testing and UI functional testing
- Tools: Selenium WebDriver, PyUnit, Pivotal Tracker and Issue tracker/ Slack
- Metrics: a) testing coverage: all completed user story features
  b) number of tests pass: 15 unit testing & 10 functional testing
  d) number of bugs: 5 (excellent work from developers)

Testing Cases

Unit testing:

--- Three test classes: TestInfrastructure(), TestAPI(), and TestSocketIO()

- ```
  def test_get_users(self):
          res = requests.get('http://localhost/api/users/?format=json')
          users = res.json()
          self.assertTrue(res.status_code == 200)
          self.assertTrue( len(users) > 0 )
  ```

- ```
  def test_global_namespace_connect(self):
          with SocketIO('localhost', 3000) as socket:
                  global_ns = socket.define(BaseNamespace, '/')
                  global_ns.emit('user', {'username':'test', 'action':'connect'})

                  res = requests.get('http://localhost:3000/users')
                  self.assertTrue('test' in res.json() )
  ```

Testing cases, cont.

UI functional testing

- previous work
  - Browser Compatibility
  - Responsive UI test
  - Button Group test
  - GeneralMessageSending test
  - GroupMessageSending test

- Current work
  - Emoticon test
    - drop-up list and emoji icons
    - emoji string name in text bar
    - emoji img in message content

Testing cases, cont.

- Current work, cont.
  - Message search test
    - searching buttons and pop-up searching results dialogue
    - different inputs and outputs
  - Upload file test
    - related buttons and pop-up dialogue
    - whole system accessible
    - file type support
  - User online status test
    - color change of user icon
  - Create new room test (show as example later)

Testing cases, cont.

UI functional testing

- Browser Compatibility
- Responsive UI test
- Button Group test
- GeneralMessageSending test
- GroupMessageSending test
- Emoticons test
- Message Search test
- Upload File test
- Create New Chatting Room test
- User Online Status test

```
def test_emoticons(self):
    driver=self.driver
    driver.get("http://localhost")
    driver.find_element_by_id("menu_button").click()
    driver.find_element_by_id("room-8").click()
    driver.find_element_by_id("emoji").click()
    driver.find_element_by_xpath("//a[@href='#2']").click()
    WebDriverWait(driver,3).until(
            EC.text_to_be_present_in_element_value((By.ID,"text"),"::unhappy::")
        )
    driver.find_element_by_id("sendcleartext").click()
    WebDriverWait(driver,3).until(
            EC.presence_of_element_located((By.XPATH,"//div[@id='room-8']/img[@src='/static/emoji/unhappy
        )
```

Testing cases, cont.

UI functional testing

- Browser Compatibility
- Responsive UI test
- Button Group test
- GeneralMessageSending test
- GroupMessageSending test
- Emoticons test
- Message Search test
- Upload File test
- Create New Chatting Room test
- User Online Status test

Test name: Create New Team testing
Test items: “create new teams” button, pop-up dialogue, team name input area, “cancel” button, “save” button, “Chatting Room” list
Test priority: high
Preconditions: login to the system
input data:

- Button click: “create new teams” button, “cancel” button, “save” button
- Input text: team name

Test steps:

- Click the “create new teams” Button
- Check the pop-up dialogue
- Input team name or input nothing
- Click the “cancel” Button or “save” button
- Check the Chatting Room list

Postcondition: the successfully created room can be clicked and entered
Expected output:

- After clicking “create new teams” button, the “Create New Team” pop-up dialogue will display and other areas on UI will become disabled
- If nothing input, then click “save” and “cancel” button, no new room will show in “Chatting Room” list
- If input team name, then click “save”, the new team name will show in “Chatting Room” list
- If input team name, then click “cancel”, the no new room will show in “Chatting Room” list

Actual output: same with expected
Pass or Fail: pass

Risk Management

Integrate project with other groups.

- Switched operating system as early as possible.
- Integrated user model as early as possible.
- Communicated with other groups and project leaders.

Risk Management

Multiple programming languages (Node.js and Django). Other teams are not familiar with Node.js.

- Used a REST API to standardize the communication between Node.js and Django
- Used object oriented principles to “hide” the details of Node.js

Risk Management

Most of the team members were unfamiliar with Django, Node.js, and javascript.

- Code review and technical mentoring by more experienced team members.
- Team members spent time learning new software

Achievements

- user friendly interface
- low latency
- integrated with other teams
- uses REST API to update client without a screen refresh
- works on both desktop and mobile

Challenges

- Group Project - communicating with team members and working as a team.
- Multi-group project
- Integrating user model
- Low latency
- Implementing Django REST API
  - Serializing objects
  - filter and search functions

Quality Management

Structural Quality:

code meets requirement of testability,maintainability,understandability, efficiency,security

Functional Quality:

meeting specified requirements.

creating software that has few defects.

good enough performance.

Process Quality:

using Agile method

software achieved on time.

Self-Evaluation

Groups cooperate with each other well.

Everyone attends group meeting at least once and knows individual task per week.

We listened to each other's ideas.

Time Limited

did not finish all the user stories.

Weak time management

Ineffectively Scheduling Tasks.

Relaxed at beginning and time limited in the end.

References

Bootstrap: http://getbootstrap.com/

SocketIO: http://socket.io/

NodeJS: http://nodejs.org/

Django: http://www.djangoproject.com/

DRF: http://www.django-rest-framework.org/