Table of Contents

Introduction

The methodology used to develop this project is **Scrum**.

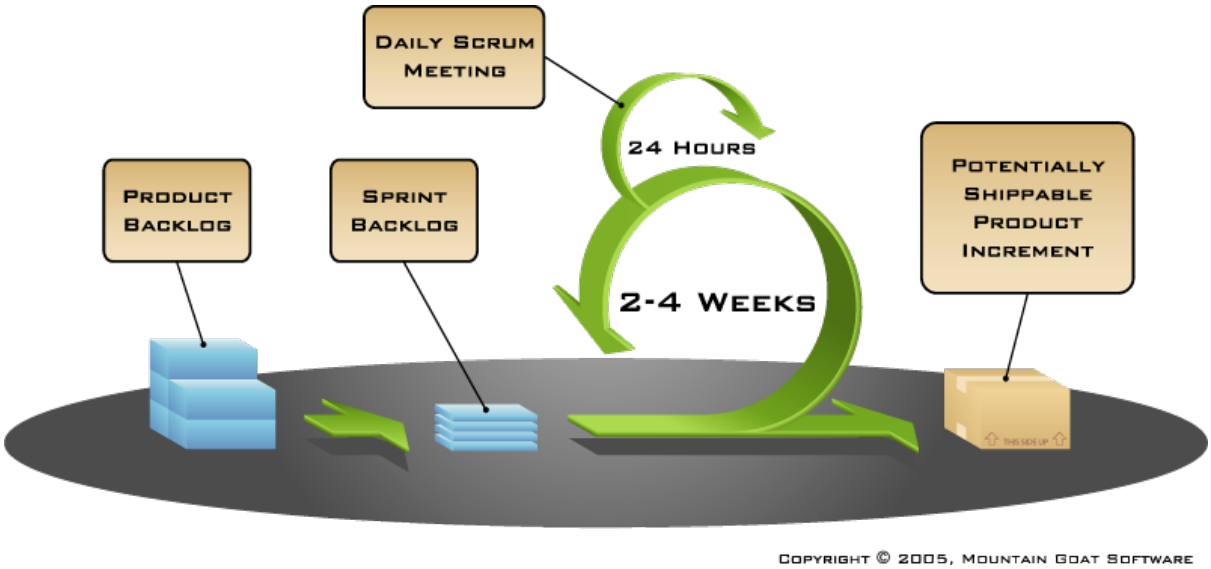

Figure 1. Scrum overview by Mountain Goat Software

Scrum is an agile process most commonly used for product development, especially software development. Scrum is a project management framework that is applicable to any project with aggressive deadlines, complex requirements and a degree of uniqueness. In Scrum, projects move forward via a series of iterations called sprints. Each sprint is typically two to four weeks long.

Scrum defines the following concepts:

- **Scrum team**: A typical scrum team has between five and nine people, but Scrum projects can easily scale into the hundreds. However, Scrum can easily be used by one-person teams and often is. This team does not include any of the traditional software engineering roles such as programmer, designer, tester or architect. Everyone on the project works together to complete the set of work they have collectively committed to complete within a sprint. Scrum teams develop a deep form of camaraderie and a feeling that “we’re all in this together.”
- **Product owner**: The product owner is the project’s key stakeholder and represents users, customers and others in the process. The product owner is often someone from product management or marketing, a key stakeholder or a key user.

- **Scrum Master**: The Scrum Master is responsible for making sure the team is as productive as possible. The Scrum Master does this by helping the team use the Scrum process, by removing impediments to progress, by protecting the team from outside, and so on.
- **Product backlog**: The product backlog is a prioritized features list containing every desired feature or change to the product. Note: The term “backlog” can get confusing because it’s used for two different things. To clarify, the product backlog is a list of desired features for the product. The sprint backlog is a list of tasks to be completed in a sprint.
- **Sprint planning meeting**: At the start of each sprint, a sprint planning meeting is held, during which the product owner presents the top items on the product backlog to the team. The Scrum team selects the work they can complete during the coming sprint. That work is then moved from the product backlog to a sprint backlog, which is the list of tasks needed to complete the product backlog items the team has committed to complete in the sprint.
- **Daily Scrum**: Each day during the sprint, a brief meeting called the daily scrum is conducted. This meeting helps set the context for each day’s work and helps the team stay on track. All team members are required to attend the daily scrum.
- **Sprint review meeting**: At the end of each sprint, the team demonstrates the completed functionality at a sprint review meeting, during which, the team shows what they accomplished during the sprint. Typically, this takes the form of a demonstration of the new features, but in an informal way; for example, PowerPoint slides are not allowed. The meeting must not become a task in itself nor a distraction from the process.
- **Sprint retrospective**: Also at the end of each sprint, the team conducts a sprint retrospective, which is a meeting during which the team (including its ScrumMaster and product owner) reflect on how well Scrum is working for them and what changes they may wish to make for it to work even better.

Analysis

The software requirements are defined through Scrum's **user stories**.

User stories are short, simple descriptions of a feature told from the perspective of the person who desires the new capability, usually a user or customer of the system. They typically follow a simple template:

> As a <type of user>, I want <some goal> so that <some reason>.

User stories are often written on index cards or sticky notes, stored in a shoe box, and arranged on walls or tables to facilitate planning and discussion. As such, they strongly shift the focus from writing about features to discussing them. In fact, these discussions are more important than whatever text is written.

User stories

#1. As a user, I want to authenticate, so that I can access protected resources

The authentication endpoint must enable a client application to login using credentials such as username and password. It should be RESTful, i.e.: it should consume and produce JSON over HTTP. An example request is like:

Listing 1. User Story {us}: authentication endpoint.

```
POST /login HTTP/1.1
Host: api.example.com
Content-Type: application/json
{
    "username": "admin",
    "password": "secret"
}
```

If the authentication is successful, the response should be like:

Listing 2. User Story {us}: successful authentication response.

```
200 OK HTTP/1.1
Content-Type: application/json
{
   "username": "admin",
   "roles": [
      "ROLE_ADMIN",
      "ROLE_USER"
   ],
   "expires_in": 3600,
   "token_type": "Bearer",
   "refresh_token":
"eyJhbGciOiJSU0EtT0FFUCIsImVuYyI6IkEyNTZHQ00ifQ.fUaSWIdZakFX7CyimRIPhuw0sfevgmwL2x
zm5H0TuaqwKx24EafCO0TruGKG-lN-
wGCITssnF2LQTqRzQGp0PoLXHfUJ0kkz5rBl6LtnRu7cdD1ZUNYXLJtFjQ3IATzoo15tPafRPyStG1Qm7-
1L0VxquhrLxkkpti0F1_VTytZAq8ltFrnxM4ahJUwS7eriivvdLqmHtnwuXw0kBXEseIyCkiyKklWDJAcD
_P_gHoQJvSCoXedlr7Pp0n6LEUrRWJ2Hb-
Zyt9dWqWDxm9nyDeEVtEZGcQtpgCGgbXxaUpULIy5nvrbRzXSNyT6iXhK1CLqiFVkfh-Y-
DHXdB6Q4sg.uYdpxl835KnlkqC5.gBgSnPWZOo6FINovJNG7Xx2RuS09QJbU4-_J4EgZQkygt8xE-
HfdYaOmtmJLjGJR1XKoaRsuX1gNjFoCZgqWAon6.Zsrk52dkjskSVQLXZBQooQ",
   "access_token": "eyJhbGciOiJSU0EtT0FFUCIsImVuYyI6IkEyNTZHQ00ifQ.n-
gGe65x0SlSXS3fTG8ZLdXvv6b5_1pDvkcGyCjFy-
vm1VhaBEQL5p3hc6iUcCAcuyrqzGk95lV9dHCv46cNfCiUFHWfbEcd4nqScIxBbc28xO9L1mNLnZ0G1rx1
Mx1L0Y_ZPoSxDXpJaHCT28cdZffHLxx2B9ioIClgdlYBAJ5Oz8VT39-
D0QSomS6QhFqmcpbDsXrsKxs545Pn-TIlu-fSQ4wpIvAxusOKB6CV2EYKqBplMBrh-
3btE8WksVcX2N3LsrcMhrKxSKi93c06MZh6JzSLWe5bl9hvUvBdEuwDrk-
fQgD3ZlmjjoevRWYhv_kslW1PlqUHYmKOQ7csUw.3mvvsFWikEjZzExA.YixjnnzzcPRy_uUpgPv5zqOfs
hv3pUwfrME0AijpsB7u9CmJe94g6f2y_3vqUps-5weKKGZyk3ZtnwEbPVAk9-HZt-
Y27SbZl4JNCFEOLVsMsK8.h4j9BdFXuWKKez6xxRAwJA"
}
```

If the authentication fails, the response should be a `401` HTTP status code.

If the request is invalid, the response shall be a `400` HTTP status code.

#2. As a user, I want to logout, so that my token is no longer valid

#3. As a developer, I want to configure how access tokens are generated

#4. As a developer, I want to store tokens on multiple storages, including the client itself

#5. As a user, I want to validate whether my access token is still valid or not

#6. As a developer, I want to expose CORS headers in the responses, so that the system can be used from separated front-end applications

#7. As a user, I want to authenticate using external OAuth providers such as Google or Facebook

Design

Development

Deployment