

FLOSScaster Project Demonstration

Brychcy, Patryk & Filatoff, Michael & Fluegel, Dwipa & Gavrilov,
Sascha & Karaman, Deniz & Lepore, Dominik & Zimmermann,
Alwin & Zundel, Benedikt

Frankfurt University of Applied Sciences

Friday, 2025-05-23

Agenda

Projektmanagement mit Scrum & GitHub Projects (Deniz Karaman)

Mockup-Design und Gestaltung (Patryk Brychcy & Michael Filatoff)

Technische Dokumentation

- Frontend (Michael Filatoff)

 - Styling (Sascha Gav)

- Backend (Benedikt Zundel)

- Dockerization (Benedikt Zundel)

- Deployment (Benedikt Zundel)

- RSS-Feed Generator (Alwin Zimmermann)

- Publikation auf einem Social Media Feed (Dwipa Flügel)

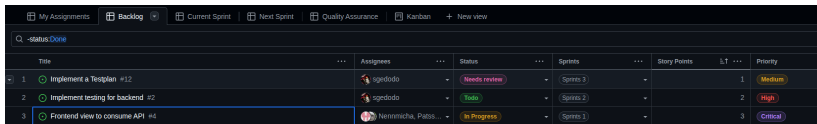
Testbericht (Dominik Lepore)

- Static Code-Analyse (Deniz Karaman)

Projektmanagement mit Scrum & GitHub Projects (Deniz Karaman)

Projektmanagement mit Scrum & GitHub Projects

Figure: Screenshot der GitHub Projects Ansicht des Backlogs

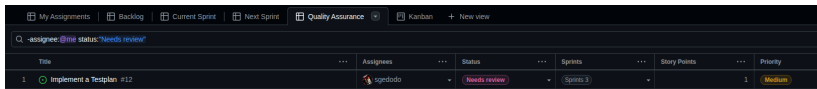


The screenshot shows the GitHub Projects interface in the 'Backlog' view. At the top, there are tabs for 'My Assignments', 'Backlog', 'Current Sprint', 'Next Sprint', 'Quality Assurance', and 'Kanban'. Below the tabs is a search bar with the text '-status Done'. The main content is a table with the following columns: Title, Assignees, Status, Sprints, Story Points, and Priority. There are three items in the backlog:



	Title	Assignees	Status	Sprints	Story Points	Priority
1	Implement a Testplan #12	sgedodo	Needs review	Sprints 3	1	Medium
2	Implement testing for backend #2	sgedodo	Todo	Sprints 2	2	High
3	Frontend view to consume API #4	Neinmicha, Patss...	In Progress	Sprints 1	3	Critical

Projektmanagement mit Scrum & GitHub Projects

Figure: Screenshot der GitHub Projects Ansicht der Quality Assurance

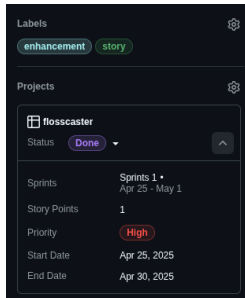


The screenshot displays the GitHub Projects interface for a project named 'Quality Assurance'. The top navigation bar includes tabs for 'My Assignments', 'Backlog', 'Current Sprint', 'Next Sprint', 'Quality Assurance' (selected), 'Kanban', and '+ New view'. Below the tabs is a search bar with the text '- assignee: @me status: "Needs review"'. The main content area shows a table with the following columns: Title, Assignees, Status, Sprints, Story Points, and Priority. A single task is listed:

Title	Assignees	Status	Sprints	Story Points	Priority
1  Implement a Testplan #12	 sgedodo	Needs review	Sprints 3	1	Medium

Projektmanagement mit Scrum & GitHub Projects

Figure: Screenshot der GitHub Ansicht einer Story



Mockup-Design und Gestaltung (Patryk Brychcy & Michael Filatoff)

Mockup-Design und Gestaltung

Figure: Figma Design der Landingpage

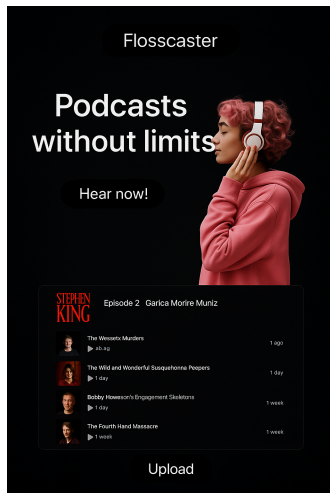
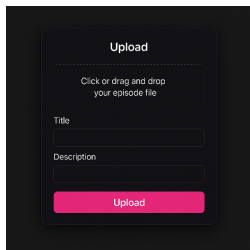


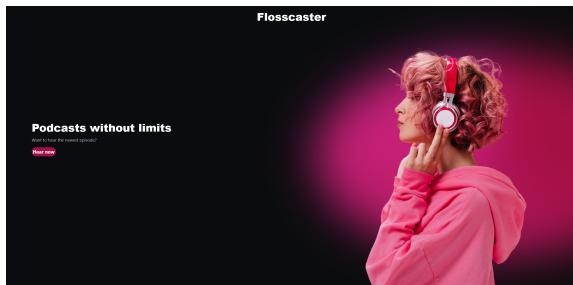
Figure: Figma Design des Upload-Dialogs



The image shows a Figma design of an 'Upload' dialog box. The dialog is a light gray rounded rectangle with a thin border. At the top, the word 'Upload' is centered in a bold, dark gray font. Below it, a dashed horizontal line separates the title from the instructions. The instructions, 'Click or drag and drop your episode file', are centered in a smaller, dark gray font. Below the instructions, there are two text input fields. The first is labeled 'Title' and the second is labeled 'Description'. Both labels are in a small, dark gray font. At the bottom of the dialog, there is a large, rounded rectangular button with a bright orange background and the word 'Upload' in white, bold, sans-serif font.

Mockup-Design und Gestaltung

Figure: Implementiertes Design der Landingpage



Mockup-Design und Gestaltung

Figure: Implementiertes Design der Podcast-Episoden



Technische Dokumentation

Frontend (Michael Filatoff)

Listing 1: Implementation des Hero-Bilds

```
1 
```

Listing 2: Implementation der Navbar

```
1 <a href="/" className={styles.mainHeader}>  
2   Flosscaster  
3 </a>
```

Styling (Sascha Gav)

Styling

Backend (Benedikt Zundel)

Listing 3: Implementation der Podcast-Episoden Klasse

```
1 @dataclass
2 class Podcast:
3     id: int
4     title: str
5     description: str
6     date: str
7     filepath: str
```

Backend

- ▶ `/api/list`
- ▶ `/api/create`
- ▶ `/api/get_upload/<path>`
- ▶ `/rss`

Dockerization (Benedikt Zundel)

Dockerization

Deployment (Benedikt Zundel)

Deployment

RSS-Feed Generator (Alwin Zimmermann)

Listing 4: Feedparser Bug

```
1 def load_existing_feed(self):
2     """Laedt den bestehenden RSS-Feed, falls vorhanden."""
3     if os.path.exists(self.feed_file_path):
4         feed = feedparser.parse(self.feed_file_path)
5         for entry in feed.entries:
6             item = {
7                 'title': entry.title,
8                 'description': entry.description,
9                 'date': entry.published if 'published' in entry else
10                 datetime.now().isoformat(),
11                 'enclosure_url': entry.enclosure.href if 'enclosure' in entry
12                 else None,
13                 'enclosure_type': entry.enclosure.type if 'enclosure' in entry
14                 else None,
15                 'enclosure_length': entry.enclosure.length if 'enclosure' in
16                 entry else None
17             }
18             self.items.append(item)
19             print(self)
```

RSS-Feed Generator

Listing 5: RSS-Feed Generator imports

```
1 import os
2 import argparse
3 import datetime
4 import pytz
5 from lxml import etree
```

Listing 6: Erstellen eines neuen Feed-Items

```
1 # Erstelle ein neues Item
2 new_item = etree.Element('item')
3 etree.SubElement(new_item, 'title').text = title
4 etree.SubElement(new_item, 'description').text = description
5 etree.SubElement(new_item, 'pubDate').text = new_episode_pubDate
6 enclosure = etree.SubElement(new_item, 'enclosure')
7 enclosure.set('url', url)
8 enclosure.set('length', length)
9 enclosure.set('type', 'audio/mpeg')
```

Publikation auf einem Social Media Feed (Dwipa Flügel)

Publikation auf einem Social Media Feed

Listing 7: Vollständige Funktion des *Autotootings*

```
1 def masttoot(title: str, url: str, description: str):
2     """Publishes a new toot using the input, also sends a reply to the
3         announcement with the description."""
4
5     post_text = "The Flosscasters have released a new podcast episode: " +
6         title + ". Check it out @ " + url
7
8     #Mastodon-Instanz
9     mastodon = Mastodon(
10         client_id = os.getenv("MASTODON_CLIENT_KEY"),
11         client_secret = os.getenv("MASTODON_CLIENT_SECRET"),
12         access_token = os.getenv("MASTODON_ACCESS_TOKEN"),
13         api_base_url = os.getenv("MASTODON_BASE_URL")
14     )
15
16     #Publish the toot and reply to it with the description
17     to_reply = mastodon.status_post(post_text)
18     mastodon.status_post(description, in_reply_to_id=to_reply.id)
```

Testbericht (Dominik Lepore)

Listing 8: Beispiel einer Fixture

```
1 @pytest.fixture()
2 def init_testing():
3     app.config.update({'TESTING': True})
4     if not os.path.exists(DATABASE_FILE):
5         con = sqlite3.connect(DATABASE_FILE)
6         cur = con.cursor()
7         cur.execute("CREATE TABLE IF NOT EXISTS podcasts(id INTEGER PRIMARY
8             KEY, title, description, date, filepath)")
9         con.close()
10    yield app
11    app.config.update({'TESTING': False})
```


Listing 9: Test-Implementation von /api/create

```
1 @pytest.mark.parametrize("title, description, audio", [  
2     ("First FLOSScast", "The very first flosscast", generate_mp3_tts("Hello to  
3     the very first flosscast episode.", lang="en")),  
4     (generate_random_ascii(length=10), generate_random_ascii(length=99),  
5     generate_mp3_size(1024)),  
6     (generate_random_ascii(length=24, letters=False, numbers=False,  
7     punctuation=True), generate_random_ascii(length=240, punctuation=True),  
8     generate_mp3_duration(30))  
9 ]) )  
10  
11 def test_create_podcasts(client, title, description, audio):  
12     response = client.post("/api/create", content_type="multipart/form-data",  
13     data={  
14         "title": title,  
15         "description": description,  
16         "audio": open(audio, 'rb')  
17     })  
18     assert response.status_code == 200
```

Listing 10: Implementation der client-Fixture

```
1 @pytest.fixture()
2 def client(init_testing):
3     with app.test_client() as client:
4         yield client
```

Listing 11: Test-Implementation von /api/list

```
1 def test_list(database_entry_id, client):
2     response = client.get("/api/list")
3     assert response.status_code == 200
4     assert f"\nid\:{database_entry_id}".encode("utf8") in response.data
5     assert b"\ntitle\:" in response.data
6     assert b"\ndate\:" in response.data
7     assert b"\ndescription\:" in response.data
8     assert b"\nfilepath\:" in response.data
```

Listing 12: Test-Implementation von /api/list gegen eine leere Datenbank

```
1 def test_list_empty_db(empty_database, client):
2     response = client.get("/api/list")
3     assert response.status_code == 200
4     assert b'[]' in response.data
```

Listing 13: Test-Implementation von /api/get_upload

```
1 def test_getFileByFilename(upload_file, client):  
2     response = client.get(f"/api/get_upload/{upload_file}")  
3     assert response.status_code == 200  
4  
5     assert "audio/mpeg" in response.headers["content-type"]
```

Listing 14: Test-Implementation von /rss

```
1 def test_getrss(create_rss, client):  
2     response = client.get("/rss")  
3     assert "application/xml" in response.headers["content-type"]  
4     assert response.status_code == 200 or response.status_code == 304
```

Listing 15: Fixture zum löschen des RSS-Feeds

```
1 @pytest.fixture()
2 def delete_rss(init_testing):
3     rss_path = os.getenv("RSS_FILE")
4     if os.path.exists(rss_path):
5         file = open(rss_path, "r")
6         rss_file = file.read()
7         file.close()
8         os.remove(rss_path)
9         return rss_file
10    return "failure"
```

Listing 16: Test-Implementation von /rss bei nicht vorhandenem Feed

```
1 def test_getrss_no_file(delete_rss, client):  
2     response = client.get("/rss")  
3     assert response.status_code == 404  
4     if not delete_rss == "failure":  
5         rss_path = os.getenv("RSS_FILE")  
6         with open(rss_path, "w") as file:  
7             file.write(delete_rss)
```


Static Code-Analyse (Deniz Karaman)

Static Code-Analyse