| **Michele Lo franco** | **Massaro Lorenzo** | **Lattarulo Claudia** |
| --- | --- | --- |

**PROJECT NAME BugBoard 26’**

| TIMING Oct 13, 2025 to DateSTATUS IN PROGRESSOWNERS [Michele Lo Franco](mailto:michelelofranco02@gmail.com) | TABLE OF CONTENTS[ABOUT US](https://docs.google.com/document/d/1XmDWLPu5D8hqP9IQVCpsA5Mj68ycefe567ogkS_l4rY/edit?tab=t.k6r4psia8xa4)[PROJECT DETAILS](https://docs.google.com/document/d/1XmDWLPu5D8hqP9IQVCpsA5Mj68ycefe567ogkS_l4rY/edit?tab=t.jnm0ecndneeb)[TRACKING](https://docs.google.com/document/d/1XmDWLPu5D8hqP9IQVCpsA5Mj68ycefe567ogkS_l4rY/edit?tab=t.mlecgsu67aoo)[RESOURCES](https://docs.google.com/document/d/1XmDWLPu5D8hqP9IQVCpsA5Mj68ycefe567ogkS_l4rY/edit?tab=t.4zbgarlknbs6) |
| --- | --- |

# PROJECT OVERVIEW

BugBoard26 nasce come un sofware per facilitare la gestione delle issues, offrendo una interfaccia che consente di segnalare, monitorare e risolvere gli errori e problematiche di sviluppo per un team di sviluppo.

| RESOURCES | |
| --- | --- |
| WIKI | File |
| TRACKER | File |
| TIMELINE | File |
| MEETING NOTES | File |

1 Glossario

| **Termine** | **Definizione** |
| --- | --- |
| Issue | Segnalazione di un problema |
| Bug | Tipo di issue che causa malfunzionamento |
| Feature | Richiesta di una nuova funzionalità |
| Admin | Utente con privilegi massimi |
| Stakeholder | Utente esterno con accesso in sola lettura |
| Mock-up | Prototipo visuale dell’interfaccia utente |
| Dashboard | Pannello di controllo |

2 Documento dei requisiti del software

2.1 Modello funzionale

2.1.1 Requisiti funzionali

* Autenticazione sicura con email e password.
* Creazione di issue con titolo, descrizione, tipo, priorità e foto.
* Assegnazione dei bug da parte degli admin.
* Esportazione dei bug in CVS, Excel e PDF.
* Assegnazione automatica dei bug.
* Dashboard amministrativa con metriche aggregate.
* Ricerca per parole chiavi.

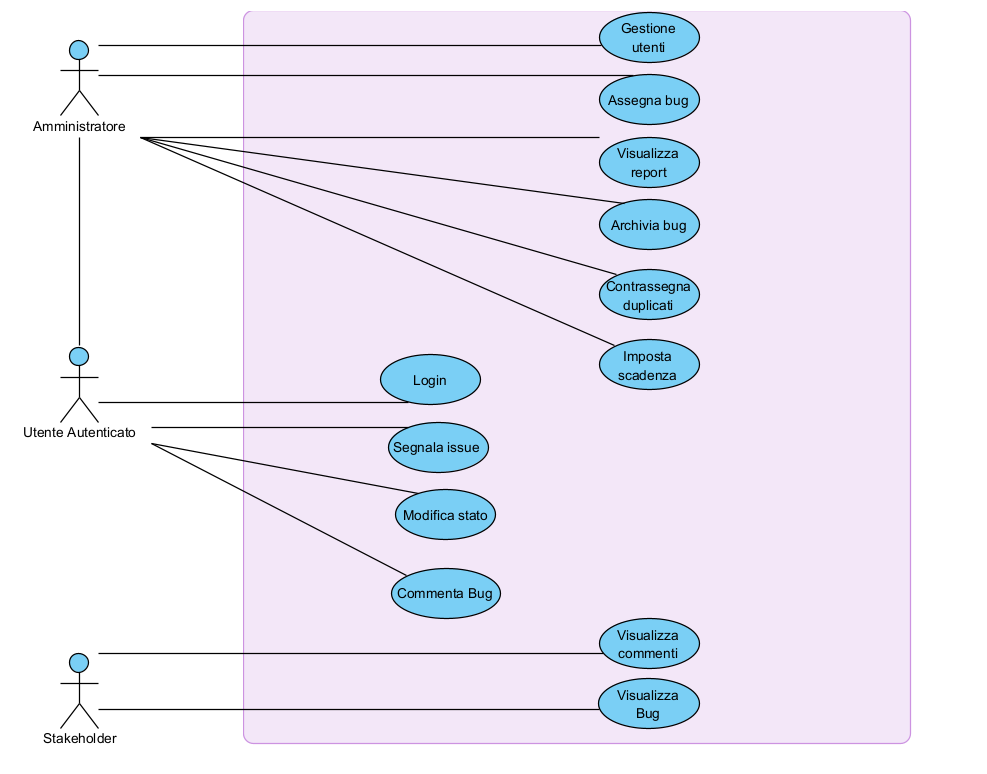
2.1.2 Requisiti non funzionali

* **Sicurezza**: protezione dei dati sensibili.
* **Usabilità**: interfaccia intuitiva e accessibile.
* **Scalabilità**: supporto a team numerosi.
* **Portabilità**: accessibile via web, mobile, desktop.
* **Indipendenza**: front-end e back-end separati.

2.2 Modellazione dei casi d’uso

Per la realizzazione dei casi d’uso abbiamo utilizzato: Visual Paradigm

2.2.1 Modellazione di uso generale:



In particolare questo use case diagram rappresenta in generale le funzionalità per ogni attore; Amministratore, Utente autenticato ed infine lo Skateholder.

| **Attore** | **Descrizione** |
| --- | --- |
| Amministratore | Gestisce gli utenti, assegna bug, visualizza report |
| Utente autenticato | Segnala bug, commenta, modifica lo stato |
| Skateholder | Visualizza bug e commenti in sola lettura |

2.3 Tabella di Cockburn

Le tabelle di Cockburn (create da Alistair Cockburn) sono uno strumento fondamentale per la rappresentazione dei casi d'uso, serve a rappresentare passo dopo passo come un attore interagisce con il sistema per raggiungere un obiettivo specifico. Questa tabella aiuta:

1. Gli sviluppatori a comprendere il comportamento atteso del sistema
2. I progettisti a definire l’interfaccia utente e le funzionalitá
3. I tester a identificare i percorsi da verificare
4. Gli skateholder a validare i requisiti funzionali

Ogni riga della tabella rappresenta un’iterazione tra utente e il sistema, mentre le estensioni descrivono cosa accade in caso di errori.

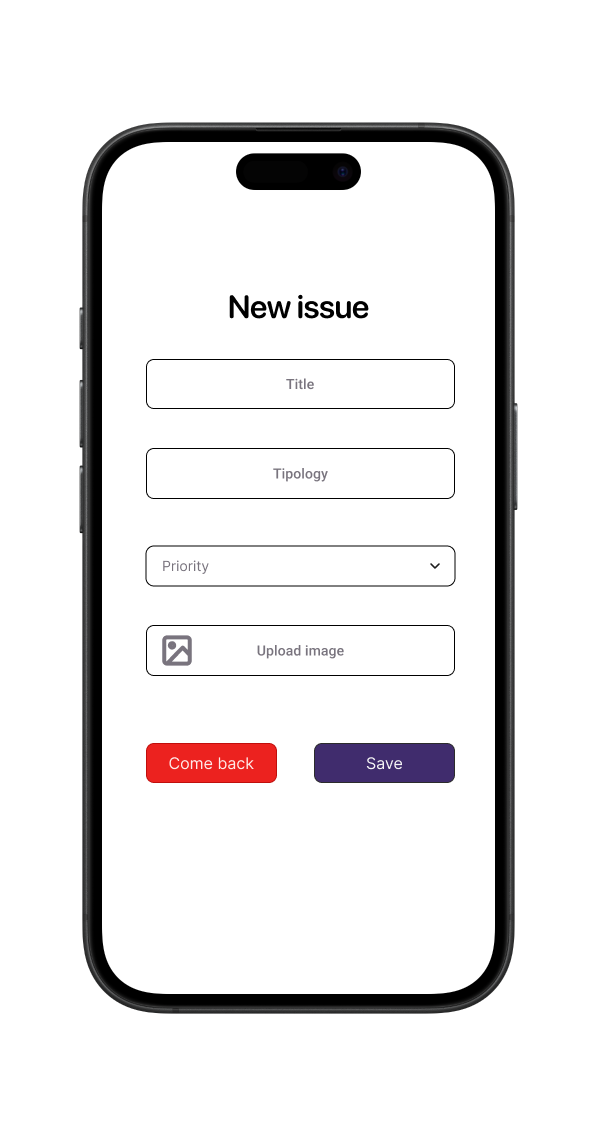
2.3.1 Segnalazione di una issue

| **Use case** | **Segnalazione di una issue** | | |
| --- | --- | --- | --- |
| **Goal in Context** | L’utente autenticato vuole segnalare una issue | | |
| **Preconditions** | L’utente è autenticato nel sistema | | |
| **Success End Condition** | L’issue viene salvata nel database con stato “todo” e visibile nella dashboard | | |
| **Failed End Condition** | Nessuna issue viene salvata; viene mostrato un messaggio di errore | | |
| **Primary Actor** | Utente autenticato | | |
| **Trigger** | L’utente clicca sul pulsante “Nuova issue” | | |
| **Main Scenario** | **Step n.** | **Utente autenticato** | **System** |
| 1 | L’utente clicca su nuova issue |  |
| 2 | Il sistema mostra il form di inserimento |  |
| 3 | L’utente compila i campi: titolo, descrizione, tipo, prioritá |  |
| 4 | L’utente allega un’immagine(opzionale) |  |
| 5 | L’utente clicca su “Salva” |  |
| 6 | Il sistema valida i dati |  |
| 7 | Il sistema salva la issue nel database |  |
| 8 | Il sistema mostra un messaggio di conferma |  |
| Extension A: Description | 7a | Il sistema rileva un errore nella validazione dei dati |  |
| 8a | Il sistema salta il salvataggio e mostra un messaggio di errore all’utente |  |
|  |  |  |  |

2.4 Mock-up

I Mock-up é una rappresentazione visiva dell’ interfaccia grafica di un’applicazione o di una funzionalitá nello specifico, serve a mostrare nel dettaglio come apparirá la schermata prima ancora di scrivere il codice.

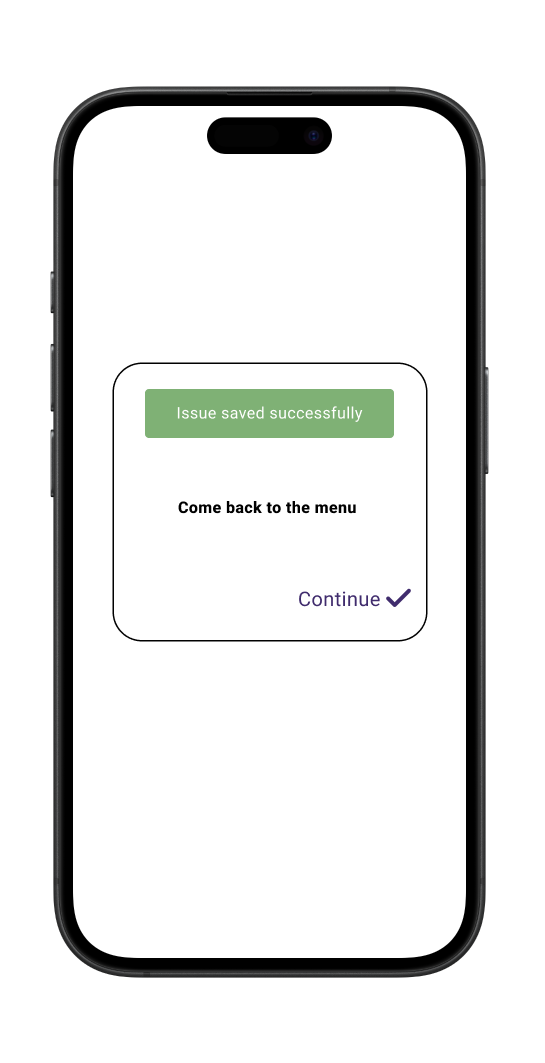
2.4.1 Inserisci nuova issue



Rappresenta la schermata iniziale dove l’utente puó:

* Inserire il titolo dell’issue
* Selezionare il tipo di issue (bug, feature, question, documentation)
* Impostare la prioritá (alta, media, bassa)
* Caricare un’immagine (opzionale)
* Cliccare su “Save” per confermare il tutto

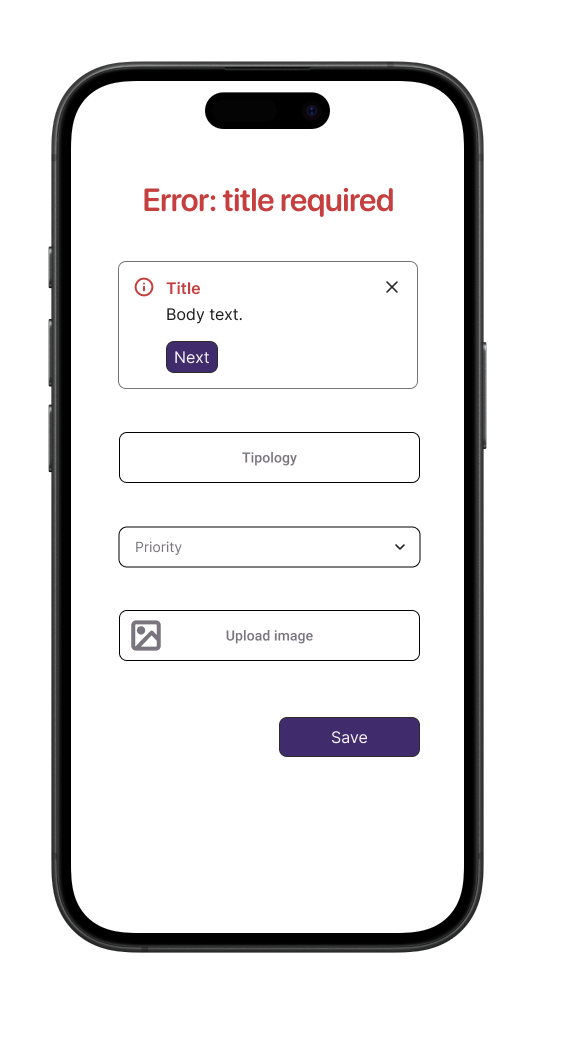
2.4.2 Messaggio di conferma



Dopo il salvataggio corretto, il sistema mostrerá:

* Una barra verde con il messaggio “Issue saved successfully”
* La dashboard si aggiornerá con la nuova issue

2.4.3 Messaggio di errore



Se l’utente dimentica di compilare un campo obbligatorio, sistema mostrerá:

* Una messaggio di errore con l’errore specificato
* Il form per la correzione