```
_={};function F(e){var t=_[e]={};return b.ea
t[1])===!1&&e.stopOnFalse){r=!1;break}n=!1,u&
?o=u.length:r&&(s=t,c(r))}return this},remove
nction(){return u=[],this},disable:function()
e:function(){return p.fireWith(this,argument
ending",r={state:function(){return n},always:
romise)?e.promise().done(n.resolve).fail(n.re
dd(function(){n=s},t[1^e][2].disable,t[2][2].
=0,n=h.call(arguments),r=n.length,i=1!==r||e&
(r),l=Array(r);r>t;t++)n[t]&&b.isFunction(n[t
/><table></table><a href='/a'>a</a><input typ
```

Hustlr - (Sprint 2)

✅ Sprint Backlog

Task	**Owner**	**Status**	**Due Date**	Story Points
Create dashboard UI	Front-End	Completed	15/04/25	5
Design UI for job posting for Clients(form)	Front-End	Completed	16/04/25	4
Design UI for job retrieval/ display for Freelancers	Front-End	Completed	16/04/25	4
Design UI for job payments for Freelancer(form)	Front-End	Completed	19/04/25	4
Design UI for payments for Clients(form)	Front-End	Completed	20/04/25	4
Add search filter for Freelancer View	Back-end	Completed	22/04/25	3
Backend endpoint for posting jobs	Back-End	Completed	21/04/25	5

Store jobs in DB	Back-End	Completed	21/04/25	3
Fetch jobs in DB	Back-End	Completed	21/04/25	3
Creation of UML diagrams	Back-End	Completed	22/04/25	2
Deploy on Azure	Front-End	Completed	23/04/25	6
UATs in correct format, code coverage tool integrated on GitHub	Back-End	Completed	23/04/25	5
25% Code coverage	Back-End	Pending	23/04/25	3
Implement Logic for Freelancer applications and validation	Team	Pending	22/04/25	5
Implement Logic for tasks, contracts, milestones.	Team	Pending	23/04/25	6
Implement Logic for “Payments”	Team	Pending	23/04/25	6

📱 Authentication & Role Management User Stories

User Story 1: Post Jobs as Client ✨

Who : Client

What: Post a job listing with details like budget, deadline, and skills required.

Why : Ensures freelancers can find and apply for my project.

Acceptance Criteria

- ✓ **Given:** the client is on the Client Dashboard. **When:** they fill in the job title, description, budget, and deadline and click "Post Job". **Then:** the job is saved and appears in the job listings for freelancers

User Story 2: Browse jobs as Freelancer 🔐

Who : Freelancer

What : Can browse and search for job listings

Why : Ensures I can find projects that match my skills.

Acceptance Criteria

- ✓ **Given:** a freelancer is logged in and navigates to the job listings page. **When:** the page loads. **Then:** they see a list of all available job listing
- ✓**Given:** a freelancer is on the job listings page. **When:** they enter a keyword, skill, or category in the search bar. **Then:** they see a filtered list of jobs matching the search criteria

User Story 3: Client access to Payments 🎯

Who : Client

What : I want to view payments I have made to Freelancers

Why : Ensures Client can keep track of payments

Acceptance Criteria

- ✓ **Given:** the client is on their dashboard. **When:** they click the "Payments" section. **Then:** they are redirected to the Payments dashboard.
- ✓**Given** the client is on the Payments dashboard. **When:** the page loads. **Then:** they see a list of all payments made to freelancers, including freelancer name, amount, date, and job title

User Story 4: Edit Jobs as Client 👨‍💼

Who : Client

What: Edit a job listing with details like budget, deadline, and skills required.

Why : Ensures freelancers can find and apply for my project.

Acceptance Criteria

- ✓ **Given:** the client is on their job listings page. **When:** they view the list of posted jobs. **Then:** they can see an "Edit" button for each job
- ✓**Given** the client clicks the "Edit" button for a job. **When:** they update the job details and click "Save". **Then:** the job listing updates immediately with the new detail.

User Story 5: Delete Jobs as Client 👨‍💼

Who : Client

What: Delete a full job listing.

Why : Ensures a client can remove a job from their listings.

Acceptance Criteria

- ✓**Given:** the client is on their job listings page. **When:** they view the list of jobs they have posted. **Then:** they see a "Delete" button next to each job
- ✓ **Given:** the client clicks the "Delete" button. **When:** they confirm the deletion. **Then:** the job is permanently removed from the listing

User Story 6: View a modern and polished UI 👨‍💼

Who : User/Page Visitor

What: I want a visually appealing landing page and other user-specific pages

Why : So I feel confident in the platform.

Acceptance Criteria

- ✓**Given:** a signed-in user accesses their dashboard or profile page.**When:** the page loads. **Then:** the user sees a modern, responsive, and visually appealing interface.
- ✓**Given:** a visitor lands on the main site (not signed in). **When:** the landing page loads.**Then:** they see a modern, responsive, and visually appealing layout

Requirements Gathering

Job Requirements

- What information is required for a job (budget, deadline, etc.)?
- Should clients vet a freelancer(receive an application), or should it be first come basis?

Payment Page Requirements

- Should admins be able to see Payments?
- What differentiates the page between a Freelancer and Client ?

📝Product Backlog

Feature/User Story	Priority	Story Points	Notes
UI for job posting (Client)	High	4	Enable clients to post job opportunities.
UI for job retrieval (Freelancer)	High	4	Display available jobs to freelancers.
UI for payment forms (Freelancer & Client)	High	8	Separate forms for both user types to manage payments.
Dashboard UI	Medium	5	Main access point for both user types.
Search filter for Freelancer jobs	Medium	3	Improve user navigation for freelancers.

Backend: job post/retrieval	High	11	Includes endpoints and DB operations.
Logic for Freelancer applications	High	5	Application and validation logic pending.
Logic for Tasks, Contracts, Milestones	High	6	Pending; critical for project tracking.
Logic for Payments	High	6	Critical functionality still pending.
Azure Deployment	Medium	6	Deployment success for testing.
UML Diagram Creation	Medium	2	Architectural clarity.
Code Coverage & UATs	High	8	Testing tools integration and tracking.

🔁Sprint Retrospective

What went well:

- UI designs for job posting, retrieval, and payments were completed efficiently.
- Backend job handling was successful and integrated with the database.
- Deployment on Azure went smoothly with minimal friction.
- UATs and GitHub code coverage tools were set up early.

What could be improved:

- Logic-heavy components (e.g. Payments, Milestones, Contracts) weren't completed as planned.
- The team underestimated the time and complexity of integrating all logic components.

Action items:

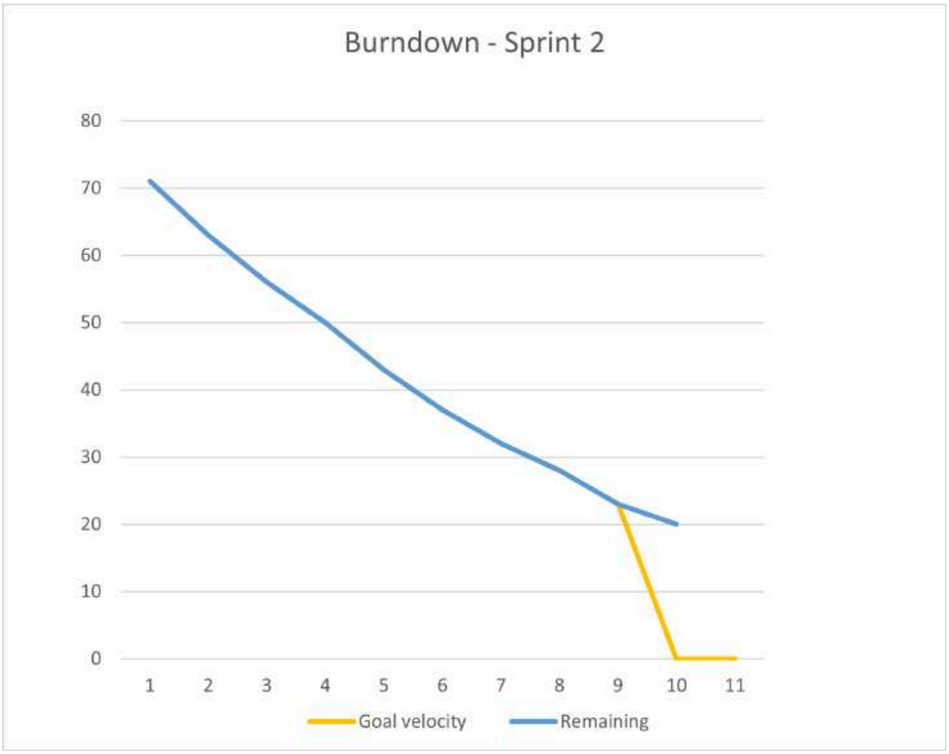
- Allocate more time for logic implementation tasks in Sprint 3.
- Introduce mini check-ins every 2 days to track progress on complex tasks.

📅Daily Stand-Up Summaries

Day	**Summary**
Day 1	Sprint planning finalized; backlog items assigned.
Day 2	Dashboard and job posting UI in progress; Azure setup revisited.
Day 3	Client & freelancer payment forms UI near completion.
Day 4	Backend endpoints for posting & retrieving jobs implemented.
Day 5	UAT test integration begun; 25% code coverage target introduced.
Day 6	Job DB storage/retrieval verified; search filter added.
Day 7	UML diagrams completed; logic for applications/payments flagged as pending.
Day 8	Final day review; logic-heavy tasks moved to next sprint; retrospective held.

📈Sprint 2 Burndown Chart

Day	Goal	Done	Goal Velocity	Remaining Effort
0	0	0	71	71
1	10	8	63	63
2	10	7	56	56
3	10	6	50	50
4	10	7	43	43
5	10	6	37	37
6	10	5	32	32
7	10	4	28	28
8	10	5	23	23
9	11	11	0	20


Burndown - Sprint 2
80
70
60
50
40
30
20
10
0
1
2
3
4
5
6
7
8
9
10
11
Goal velocity
Remaining